

Angewandte Computer- und Biowissenschaften

Professur Medieninformatik

Bachelorarbeit

Entwicklung und Implementierung eines Animationstools zur
Visualisierung katalytischer Vorgänge am Beispiel der
elektrochemischen CO₂-Reduktion

Anna Heinrich

Mittweida, den 10. Dezember 2017

Erstprüfer: Prof. Dr. Marc Ritter

Zweitprüfer: Prof. Dr. Iris Herrmann-Geppert

Co-Betreuer: Manuel Heinzig, M.Sc.

Heinrich, Anna

Entwicklung und Implementierung eines Animationstools zur Visualisierung katalytischer Vorgänge am Beispiel der elektrochemischen CO₂-Reduktion

Bachelorarbeit, Angewandte Computer- und Biowissenschaften

Hochschule Mittweida – University of Applied Sciences, Dezember 2017

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis	V
1 Einführung und Motivation	1
1.1 Rahmenprojekt BIOTACMI	2
1.2 Zielstellung und Aufbau der Arbeit	2
1.3 Das Visualisierungstool AtomCAD	3
2 Grundlagen	5
2.1 Ansatz vergleichbarer Applikationen	5
2.1.1 2D-Zeichentools mit 3D-Viewer	6
2.1.2 3D-Lernsoftware	7
2.1.3 CellUnity	7
2.2 Die elektrochemische CO ₂ -Reduktion	7
2.2.1 Wissenschaftliche Bedeutung	8
2.2.2 Unterscheidung und Reaktionsablauf	8
2.3 Unity	10
2.3.1 Grafische Benutzerschnittstellen – GUIs	10
2.3.2 Physikalische Berechnung der Engine	11
2.3.3 Kameras und Rendering	12
2.3.4 Raycasting	14
2.4 XML	14
2.4.1 Aufbau eines XML-Dokuments	14
2.4.2 Anwendungsmöglichkeiten in Unity	15
3 Anforderungen und Spezifikation	17
3.1 Aufbau der grafischen Benutzerschnittstelle	17

3.1.1	Hauptmenü	17
3.1.2	Auswahlmenü	18
3.1.3	Szene-GUI	18
3.2	Reaktionsparameter	18
3.3	Grafische Illustrierung des Reaktionsablaufs	19
3.3.1	Darstellung chemischer Komponenten	19
3.3.2	Visualisierung der Reaktionsmechanismen	20
3.3.3	Kamerabewegung	23
3.4	Assets	23
3.4.1	3D-Modelle	23
3.4.2	Materials	24
3.4.3	Physic Materials	25
3.4.4	Grafiken	25
3.5	Usability	25
3.5.1	Definition	26
3.5.2	Angewandte Methoden	26
3.6	Alleinstellungsmerkmale	26
4	Implementierung	29
4.1	Menüs	29
4.1.1	Startmenü	29
4.1.2	Auswahlmenüs	29
4.2	Aufbau des Szene-GUIs	30
4.2.1	Menüleiste	31
4.2.2	Zeitleiste	32
4.2.3	Spawn-Menü	32
4.2.4	Baukasten-Menü	34
4.3	Reaktionsumgebung	34
4.3.1	Aufbau der Testreaktion	34
4.3.2	Rendering	36
5	Evaluation	39
5.1	Reaktionsverlauf	39

5.1.1	Funktionskontrolle	39
5.1.2	Testszenarien	41
5.2	Identifizierte Probleme	43
5.2.1	Visualisierung	43
5.2.2	Kollisionserkennung	43
5.2.3	Physikalische Verhaltensweisen	44
5.2.4	Performanz	44
6	Zusammenfassung	47
6.1	Optimierungsmöglichkeiten	47
6.1.1	Kollisionserkennung	47
6.1.2	Physikalische Verhaltensweisen	48
6.2	Ausblick	48
6.2.1	Parameter	49
6.2.2	Komponenten	49
6.2.3	Funktionen	51
	Literaturverzeichnis	IX
	Anhang	XIII

Abbildungs- und Tabellenverzeichnis

Abb. 2.1	Reaktionsablauf der elektrochemischen CO ₂ -Reduktion	9
Abb. 2.2	Rendering einer Beispielszene mit verschiedenen Kamera- Einstellungen	13
Abb. 2.3	Beschreibung von Wasserstoff in XML	15
Abb. 3.1	Molekül-Baukasten	22
Abb. 4.1	Use-Case-Diagramm der Haupt- und Auswahlmenü-Struktur . .	30
Abb. 4.2	Szene-GUI	31
Abb. 4.3	Spawn-Panel	32
Abb. 4.4	Use-Case-Diagramm des Haupt-GUIs	35
Tab. 5.1	Systeminformationen	40
Tab. 5.2	Analyse von zwölf Testszenarien (1)	42
Abb. 6.1	Cobalt-Tetramethoxyphenylporphyrin	51

1. Einführung und Motivation

Mit einer sich immer weiter digitalisierenden Gesellschaft ändert sich auch die Art und Weise, wie Wissen vermittelt wird, nachhaltig. Dieser Prozess beschränkt sich jedoch oft auf die reine Konvertierung analoger Bildungsmedien ins Digitale. Anstatt Notizen auf Papier zu schreiben, tippen Studierende sie ab, anstatt einer Tafel verwenden Dozenten digitale Whiteboards, Lehrbücher werden zusätzlich als E-Books veröffentlicht.

Obwohl Argumente für jedes der genannten Beispiele leicht zu finden sind, fällt es auch als Digital Native [Pre01, S.1] keineswegs schwer, Aversionen gegenüber „neuen alten“ Bildungsmedien nachzuvollziehen – ist doch selbst die Funktionalität von Projektoren in Hörsälen Glückssache und nur geringfügig abhängig von der Technikaffinität des Vortragenden. Wiegen Bildungsbeauftragte also die Umständlichkeit der Bedienung (ein subjektiver Faktor) gegen den Mehrwert des jeweiligen Mediums (ebenfalls subjektiv) ab, fällt die Bewertung zwangsläufig subjektiv aus.

Objektiv nicht zu bestreiten ist allerdings, welche Chancen der gezielte Einsatz digitaler Technologien für den Bildungsbereich mit sich bringen kann. Vor allen Dingen im naturwissenschaftlichen Bereich versprechen in etwa Virtual-Reality-Anwendungen, Wissenschaft erlebbar zu machen und Unterricht – sei es an Schulen oder Universitäten – zu ergänzen.

Als erster Schritt auf dem Weg, komplexes Fachwissen verständlich zu vermitteln, können für den Bildungsbereich konzipierte Programme helfen, ein Grundverständnis für Regeln und Abläufe zu schaffen und so das Interesse des Anwenders zu wecken. Idealerweise wird diesem ermöglicht, mehr Bezug zu abstrakten Forschungsgebieten zu erhalten und ihn dafür zu sensibilisieren.

1.1. Rahmenprojekt BIOTACMI

Wissen durch virtuelle Visualisierung zu vermitteln und vertiefen ist das Ziel der interdisziplinären Arbeitsgruppe BIOTACMI, die sich aus den Professuren Biotechnologie, Angewandte Chemie und Medieninformatik an der Hochschule für angewandte Wissenschaften Mittweida zusammensetzt. Im Rahmen dieser Kooperation sind mehrere Anwendungen geplant, welche einerseits Studierenden der Naturwissenschaften Lernhilfen bieten sowie andererseits Dozenten erlauben werden, ihre Vorlesungen, Übungen oder Seminare aufzuwerten.

So soll in etwa ein VR-Labor grundlegende Einführungen in die Laborpraxis gewähren, indem die Anwendung zahlreicher Methoden und Handgriffe geprobt werden kann, ohne reale Ressourcen verwenden zu müssen. Ein Beispiel hierfür stellt das Pipettieren dar, welches mithilfe von Controllern im VR-Raum simuliert wird. [\[Knu17\]](#)

Ein weiteres von BIOTACMI geplantes Projekt ist ein Programm, das die Darstellung chemischer Prozesse erlaubt und einen näheren Einblick in Reaktionsmechanismen und -abläufe gewährt. Dessen Konzeptionierung soll in dieser Arbeit beschrieben werden.

1.2. Zielstellung und Aufbau der Arbeit

Ziel dieser Arbeit ist die prototypische Entwicklung und Implementierung eines Tools, welches der Visualisierung katalytischer Vorgänge dient. Anhand der elektrochemischen CO₂-Reduktion als Anwendungsfall wird erforscht, inwiefern sich mit diesem effektiv chemische Reaktionen animieren lassen. Die Umsetzung erfolgt mit der Programmiersprache C# in der *Unity Engine*.

Dazu wird in Kapitel 2 untersucht, wie vergleichbare Anwendungen bezüglich ihrer Funktionen und ihres Designs aufgebaut sind. Es folgt eine kurze Vorstellung der zu animierenden Beispielreaktion im Hinblick auf deren wissenschaftliche Signifikanz. Ein fundierter Überblick der von der *Unity Engine* gebotenen Funktionen ist für die Implementierung unerlässlich; deswegen soll ausführlich behandelt werden, welche Möglichkeiten diese Entwicklungsumgebung bezogen auf die Gestaltung und Funktionalität des Programms bietet.

In Kapitel 3 werden darauf folgend die an das Tool gestellten Anforderungen erörtert – angefangen bei der Strukturierung der grafischen Benutzerschnittstelle über die visuelle Darstellung der Beispielreaktion bis hin zu den benötigten *Unity Assets* und relevanten Alleinstellungsmerkmalen. Im Vordergrund von Kapitel 4 steht danach die prototypische Implementierung der Kernfunktionen in *Unity*. Der Aufbau der Beispielreaktion innerhalb des Programms und die Evaluation der Funktionalität wird schließlich in Kapitel 5 beschrieben. Zuletzt bietet das 6. Kapitel neben einer Zusammenfassung der Ergebnisse einen ausführlichen Ausblick auf mögliche Ergänzungen.

1.3. Das Visualisierungstool AtomCAD

Um auf das Visualisierungs-Tool besser Bezug nehmen zu können, empfiehlt es sich, diesem vor der Konzeptionierung einen vorläufigen Arbeitstitel zuzuweisen. Ausgehend vom geplanten Funktionsumfang wurde als passende Bezeichnung „AtomCAD“ gewählt. Dadurch wird das Programm bereits als Anwendung charakterisiert, die den computergestützten Entwurf (Computer Assisted Design) von chemischen Elementen, Verbindungen und Reaktionen ermöglicht.

2. Grundlagen

Da bei der Entwicklung des Prototyps zunächst der Aufbau und die Funktionalität des User Interfaces Berücksichtigung findet, soll dieses Kapitel dazu dienen, Ansätze verschiedener Visualisierungstools im Bereich der Chemie zu erläutern sowie zu vergleichen. Die Analyse der gewählten Programme soll dazu beitragen, eine Orientierungshilfe zu schaffen, auf die in den folgenden Kapiteln im Bezug auf mögliche und benötigte Funktionen von AtomCAD zurückgegriffen werden kann.

Darüber hinaus soll die Testreaktion ausgehend von den zu Beginn vorliegenden Komponenten über mögliche Zwischenschritte bis hin zu ihren Endprodukten aufgeschlüsselt werden. So entsteht ein Leitfaden für die Planung der Reaktionsvisualisierung, der als Hilfestellung für deren Umsetzung in *Unity* dient. Um alle nötigen Aspekte berücksichtigen zu können, empfiehlt sich hier zunächst eine Betrachtung des Grundprinzips der elektrochemischen CO₂-Reduktion, deren Anwendungsgebiete sowie des aktuellen Forschungsstands.

Der vorletzte Abschnitt des Kapitels widmet sich schließlich der *Unity Engine* sowie der Vorstellung deren wichtigster sowie für die Entwicklung von AtomCAD relevanter Funktionen. Um das Tool effizient erweitern zu können, kommt die Beschreibung von Atomen und Molekülen in Textdateien infrage, die der Nutzer beliebig ergänzen kann. Ein hierfür geeignetes Format ist XML. Im letzten Abschnitt soll deswegen erörtert werden, welche Synergien zwischen XML und *Unity* bestehen.

2.1. Ansatz vergleichbarer Applikationen

Im Bereich der chemischen Visualisierung existiert eine Vielzahl von Anwendungen, die der Darstellung molekularer Verbindungen dienen. Hochspezialisierte Produkte, die beispielsweise bei der Modellierung von Proteinstrukturen zum Einsatz kommen, können im Kontext dieser Arbeit zwar außer Acht gelassen werden; trotzdem bleibt der potenzielle Nutzer mit einer umfangreichen Auswahl konfrontiert. [RCS17]

Um dennoch eine Übersicht zu gewährleisten, die nicht zu detailliert ausfällt, wurde eine Unterscheidung zwischen verschiedenen Programm-Typen gewählt, die sich in der Art ihrer Bedienung voneinander unterscheiden. So kann eine Zuordnung einiger Beispiele erfolgen, ohne zu ausführlich auf deren genauen Funktionsumfang eingehen zu müssen.

2.1.1. 2D-Zeichentools mit 3D-Viewer

Diese Typbeschreibung umfasst Anwendungen, die wie vektor- oder pixelbasierte Zeichenprogramme aufgebaut sind. Der Nutzer wählt aus einer Palette Werkzeuge und zeichnet damit auf einer zweidimensionalen Fläche die Strukturformel einer chemischen Verbindung. Aus dieser wird auf Knopfdruck automatisch ein 3D-Modell generiert, welches der Nutzer aus verschiedenen Blickwinkeln betrachten kann.

Ein Beispiel hierfür stellt *ChemSketch* von ACD/Labs dar. Es erlaubt dem Nutzer im „Structure“-Modus, Reaktionsgleichungen aufzustellen. Der „Draw“-Modus dient dem Zeichnen von Verbindungen. Deren Darstellung im 3D-Viewer lässt sich geringfügig anpassen - einstellbar ist u.a., ob einzelne Atome eines Moleküls als Kugel oder Stab erscheinen und wie Bindungen visualisiert werden. [Adv17]

Eine weitere Anwendung, die zu der Typbeschreibung passt, ist *BIOVIA Draw*. Hierbei handelt es sich um ein Teilprodukt eines Software-Gesamtpakets, welches ähnlich wie *ChemSketch* aufgebaut ist. Die Besonderheit bei diesem Programm besteht darin, dass es mit anderen Anwendungen des Software-Pakets interkompatibel ist und Inhalte zur weiteren Verwendung zwischen diesen übertragen werden können. [Das17]

Auch *MolView*, eine web-basierte Datenvisualisierungsplattform, dient dazu, Moleküle zunächst als Strukturformel zu zeichnen und diese dann in ein 3D-Modell zu konvertieren bzw. in verschiedene Formate zu exportieren, die von anderen Anwendungen geladen und weiterverarbeitet werden können. Außerdem bietet es Zugriff auf fachspezifische Datenbanken, aus denen der Nutzer eine Vielzahl an Verbindungen laden und als 3D-Modell betrachten kann. Dazu zählen Protein- und Kristallstrukturen. [Ber15]

2.1.2. 3D-Lernsoftware

Diese Typbeschreibung bezeichnet für den Bildungsbereich konzipierte Anwendungen, die sich auf grundlegender Ebene mit der 3D-Darstellung chemischer Elemente, Verbindungen sowie Reaktionen auseinandersetzen.

Ein Beispiel dafür ist *MEL Chemistry VR*. Hierbei handelt es sich um ein Virtual-Reality-Labor. Informationen werden dem Nutzer in Lektionen gegliedert präsentiert. Diese behandeln beispielsweise den Aufbau von Atomen, deren Anordnung in Ionengitterstrukturen, Aggregatzustände oder Kristallisation. Mit Quizabfragen wird getestet, ob der Nutzer das in den Lektionen vermittelte Wissen verstanden hat. [MEL17]

Labster ist eine ähnlich aufgebaute VR-Anwendung. Die dafür angebotenen Simulationen sind thematisch im naturwissenschaftlichen Bereich angesiedelt, beschränken sich also nicht nur auf chemische Verbindungen und Reaktionen. [Lab17]

2.1.3. CellUnity

Bei *CellUnity* handelt es sich um ein interaktives Tool, welches der Visualisierung molekularer Reaktionen dient. Da es die einzige identifizierte Anwendung ist, die nicht nur Verbindungen, sondern auch Reaktionsprozesse im dreidimensionalen Raum dynamisch darstellt, lässt es sich keinem anderen Typ zuordnen. Moleküle können aus Dateien vom Typ PDB (Protein Data Bank) geladen und innerhalb von *Unity* als Ausgangs- und Endprodukte einer Reaktion vordefiniert werden, welche der Nutzer im Game-Modus manuell initialisiert. Anstatt den Reaktionsablauf streng an die Prinzipien physikalischer Simulationen zu binden, ist der Reaktionszeitpunkt so abhängig von den Eingaben des Nutzers. [Geh14]

2.2. Die elektrochemische CO₂-Reduktion

Die elektrochemische CO₂-Reduktion ist das Anwendungsbeispiel, anhand dessen die spätere Evaluation der Funktionalität von AtomCAD erfolgt. Die folgenden Abschnitte sollen sich deshalb der wissenschaftlichen Bedeutung dieser chemischen Reaktion sowie deren Reaktionsablauf widmen.

2.2.1. Wissenschaftliche Bedeutung

„Bisher wird Kohlendioxid (CO_2) in der Öffentlichkeit vorwiegend als „Problemstoff“ im Zusammenhang mit der Erwärmung der Erdatmosphäre wahrgenommen. Dies könnte sich in Zukunft ändern. Es wird weiterhin notwendig sein, CO_2 -Emissionen so weit wie möglich zu vermeiden. [...] Darüber hinaus kann CO_2 aber auch vom Problemstoff zum Wertstoff werden und als chemischer Baustein stofflich genutzt werden.“ [Bun13, S.3]

Das grundlegende Prinzip der künstlichen Photosynthese besteht darin, aus Kohlenstoffdioxid und Wasser mittels Sonnenlicht nutzbare Kohlenwasserstoffe (wie z.B. Methan) zu synthetisieren. In der Theorie handelt es sich dabei um eine vielversprechende Methode zur Reduktion der Konzentration von CO_2 in der Atmosphäre bei gleichzeitiger Gewinnung von industriell relevanten Rohstoffen. [Dri15, KSHY15, Har15]

In ersten Ansätzen konnte zwar CO_2 erfolgreich umgesetzt werden, aber die Reaktion läuft mit relativ hohen Energieverlusten ab und führt zu einem breiten Spektrum an Reaktionsprodukten. Dies liegt in der Anzahl der notwendigen Elektronentransferschritte von der Elektrode zu den Zwischenprodukten, den hohen Konzentrationen von CO_2 an der Elektrode und der mit der CO_2 -Reduktion konkurrierenden Wasserstoffentwicklung begründet. Nach Katalysatoren, die diese Probleme überwinden und CO_2 mit hoher Selektivität und hohen Wirkungsgraden umsetzen, wird deshalb kontinuierlich geforscht [Max17].

2.2.2. Unterscheidung und Reaktionsablauf

Bei der elektrochemischen CO_2 -Reduktion handelt es sich um eine stark vereinfachte künstliche Photosynthese. In der Photosynthese wird Licht im Chlorophyll von Blättern absorbiert. Die gewonnene Lichtenergie wird genutzt, um Wasser zu oxidieren und in der Dunkelreaktion (Calvin-Zyklus) CO_2 in mehreren Schritten an Enzymen zu Glucose zu reduzieren.

Dagegen findet die Umsetzung von CO_2 im Fall der elektrochemischen CO_2 -Reduktion in einer Dunkelreaktion an Elektroden mit einer angelegten Spannung statt. Die

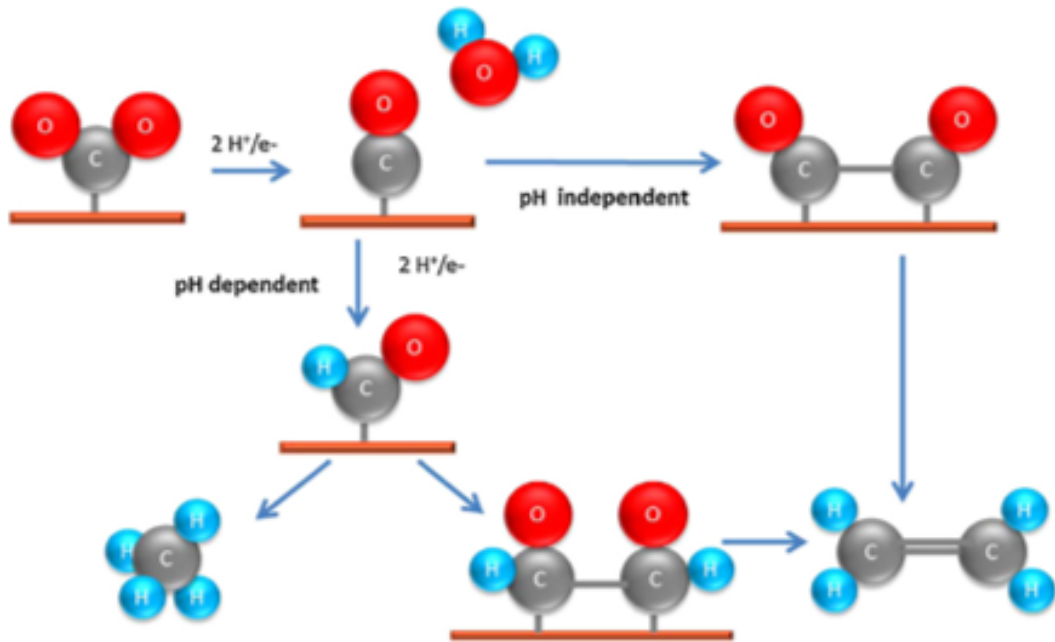


Abbildung 2.1.: Reaktionsablauf der elektrochemischen CO₂-Reduktion nach Nørskov und Koper [SVKR16, S.12]

Brennstoffherzeugung kann in einer elektrochemischen Zelle erfolgen. Die Reaktion läuft in mehreren Teilschritten ab, die auf Abbildung 2.1 illustriert sind.

Zunächst adsorbiert CO₂ an den reaktiven Zentren der Elektrode (hier oben rechts). Wenn eine Aktivierung an dem unpolaren Molekül stattfindet, werden zwei Elektronen von der Elektrode auf das Molekül übertragen. Wenn Protonen auf den gebildeten Übergangszustand treffen, kann ein Wassermolekül und Kohlenstoffmonoxid (CO) entstehen. Für diesen ersten entscheidenden Reaktionsschritt müssen sogenannte aktive Zentren auf der Elektrodenoberfläche bereitstehen, die das Adsorbieren von CO₂ ermöglichen. Wenn diese nicht zur Verfügung stehen, adsorbieren nur Protonen an der Oberfläche und es bildet sich Wasserstoff.

Der weitere Verlauf der Reaktion hängt vom pH-Wert ab. In der Literatur werden dazu mehrere Möglichkeiten diskutiert [LHZQ14, RBJ⁺17]. In der hier vorliegenden Arbeit wird das theoretisch berechnete Modell von Nørskov und Koper [NPAP⁺10, CVK13] herangezogen. Wenn der Elektrolyt sauer ist (also eine hohe Protonenkonzentration vorliegt), kann das CO mit weiteren transferierten Elektro-

nen und Protonen zu einem CHO-Übergangskomplex bis hin zu Methan (CH_4) reagieren. In neutralen oder basischen Elektrolyten können die CO-Moleküle adsorbiert an der Elektrode zu einem $(\text{CO})_2$ -Komplex reagieren, wenn die Abstände zwischen den CO-Molekülen klein genug sind und ein Mangel an Protonen vorliegt. Ein mögliches Produkt weiterer Reduktionsschritte, in denen wieder Protonen zur Verfügung stehen, ist Ethen (C_2H_4).

2.3. Unity

Die *Unity Engine* ist eine Entwicklungsumgebung, die ursprünglich für die Erstellung von Videospielen konzipiert wurde. Sie kommt bei der Implementierung von AtomCAD unter anderem deswegen zum Einsatz, da sie vielfältige Möglichkeiten der Gestaltung von grafischen Benutzerschnittstellen bietet und bereits zahlreiche vorgefertigte Klassen enthält, die zur Reduzierung des Programmieraufwands beitragen sowie die Umsetzung der benötigten Funktionen vereinfachen können.

Die Engine unterstützt darüber hinaus verschiedene VR-Devices, eine spätere Anpassung von AtomCAD für die Virtual Reality liegt damit im Bereich des Möglichen. Somit könnte auch die nahtlose Anbindung des Programms an das von BIOTACMI ebenfalls geplante VR-Labor realisiert werden.

2.3.1. Grafische Benutzerschnittstellen – GUIs

Die *Unity Engine* stellt mit ihrem UI-System ein effektives Werkzeug zur Verfügung, welches das schnelle und intuitive Design von grafischen Benutzerschnittstellen (auch Graphical User Interfaces oder kurz GUIs genannt) erlaubt [Uni17b, UI].

Hauptobjekt dieses UI-Systems ist das *Canvas*-Objekt. Bei jedem UI-Element, welches vom UI-System dargestellt werden soll, muss es sich um ein Kind-Objekt eines *Canvas* handeln. Das *Canvas* selbst wird durch ein Rechteck dargestellt, das sich dem Bildschirm abhängig vom sog. *Render Mode* selbstständig anpassen kann. Der *Render Mode* ist die wichtigste Eigenschaft eines *Canvas*, da dieser festlegt, auf welche Art es und ihm untergeordnete Objekte gerendert werden sollen. Es gibt drei Arten von *Render Modes*: Bei „Screen Space - Overlay“ passt sich das *Canvas* der Größe des aktuellen Bildschirms an und wird einschließlich seiner Kind-Objekte über

das normale Kamerabild gelegt. Dagegen ermöglicht „Screen Space - Camera“ zusätzlich die Darstellung der GUI-Elemente mit räumlicher Tiefe. Das *Canvas* verhält sich hier so, als ob es auf einer Plane, die sich in einer editierbaren Entfernung zur Kamera befindet, projiziert wird. Der letzte *Render Mode*, „World Space“, gestattet es, ein *Canvas* als *GameObject* in der Szene zu platzieren. [SW17, S.344ff]

Unabhängig von der späteren Menge und Art der in AtomCAD implementierten Menüs ist die korrekte Anordnung und Organisation von UI-Objekten eine Notwendigkeit. Hierfür bietet *Canvas* ein vielseitig anpassbares Layout-System an, über welches Position und Ausrichtung von Objekten auch unabhängig von Bildschirmgröße und Auflösung definiert werden können. [Uni17b, Basic Layout]

Zusätzliche Kontrolle über die Positionierung von UI-Elementen lässt sich auch mit deren Zuordnung zu einem sogenannten *Panel* erwirken. Bei diesem handelt es sich um eine Art Bild-Objekt, welches als Container für andere UI-Objekte dienen kann und sich damit für die Gliederung der einem *Canvas* untergeordneten Komponenten eignet [SW17, S.355]. Ein Menü, das sich in drei Untermenüs aufteilt, ließe sich mit drei *Panels* konstruieren. Das Layout der einem *Panel* zugewiesenen UI-Elemente ist relativ zu diesem, während sich das Layout des *Panels* selbst relativ zu *Canvas* verhält. Daraus ergeben sich vielseitige Möglichkeiten der Gestaltung von Graphical User Interfaces.

Neben den visuellen Komponenten, die wie *Canvas* und *Panel* Organisation, Layout und Darstellung dienen, steht auch eine Auswahl an funktionalen UI-Objekten zur Verfügung, womit an dieser Stelle alle UI-Elemente bezeichnet werden, mit denen der Nutzer direkt interagiert. Hierzu zählen *Buttons*, *Toggles*, *Slider*, *Scrollbars*, *Dropdowns* und *Input Fields*. [Uni17b, Interaction Components]

2.3.2. Physikalische Berechnung der Engine

Nach *Unity* importierte oder darin erzeugte Objekte benötigen physikalische Eigenschaften, damit sie miteinander interagieren können. Hierfür liefert die Engine eine Reihe von Komponenten mit, die die Delegation verschiedener physikalische Verhaltensweisen an Objekte erlauben. Während für die meisten dieser Komponenten ebenfalls ein 2D-Pendant existiert, soll an dieser Stelle lediglich auf die Physikberechnung im dreidimensionalen Raum eingegangen werden, da nur diese für AtomCAD Relevanz hat.

Physik-Eigenschaften werden regelmäßig in definierten Zeitabständen berechnet. Danach wird - soweit vorhanden - in allen Skripten die *FixedUpdate*-Methode aufgerufen. Das Intervall, in dem Berechnungen stattfinden, kann hierbei verändert werden, um eine bessere Kollisionserkennung zu ermöglichen oder aber die Performanz zu verbessern. [SW17, S.209f]

Kern der *Unity*-Physikberechnung ist die *Rigidbody*-Komponente. Wird sie einem Objekt hinzugefügt, beeinflusst die Physik-Engine fortan dessen Bewegung. Ein grundlegendes Beispiel ist hier die Wirkung der Schwerkraft, wobei *Rigidbody* auch andere Eigenschaften kontrolliert - es lassen sich i.e. Masse, Massenschwerpunkt und Luftwiderstand eines Objekts einstellen. Ergänzend zur Definition der physikalischen Parameter gestattet *Rigidbody* das Hinzufügen von Kräften sowie Drehmomenten. Dies kann sowohl passiv durch Kollision mit anderen Objekten als auch aktiv durch eine Zuweisung per Skript geschehen. [SW17, S.210–215]

Ein weiterer wichtiger Bestandteil der Physik-Engine ist die Kollisionserkennung, die primär dem Erkennen von Zusammenstößen unterschiedlicher Objekte dient, aber auch für das Auslösen von Ereignissen nutzbar ist. Die Hauptkomponente zum Registrieren von Kollisionen ist die *Collider*-Komponente, die den eigentlichen *Collider* erzeugt, sobald sie dem Objekt zugewiesen wird. *Collider*-Komponenten gliedern sich in vier verschiedene Typen. Bei drei davon handelt es sich um *Primitive Collider*, die durch Grundformen beschrieben werden: Quader, Kugel und Kapsel. Der sogenannte *Mesh Collider*, der sich der Form des jeweiligen Objekts anpasst, dem er hinzugefügt wird, ergänzt diese Auswahl. [SW17, S.215ff]

2.3.3. Kameras und Rendering

Ein zentraler Aspekt jeder Visualisierung ist der Blickwinkel des Betrachters zum Geschehen. Diesen steuert in *Unity* die *Camera*-Komponente. Deren Anpassung ist mittels zahlreicher Einstellungen möglich. Anhand von Abbildung 2.2 lässt sich der Einfluss der wichtigsten Parameter erläutern.

Sowohl die neun farbigen Würfel als auch das Kamera-Objekt befinden sich durchgehend an einer festen Position. Das erste Bild zeigt die unveränderte Kameraperspektive. Alle Würfel sind sichtbar, im Hintergrund wird die Standard-Skybox

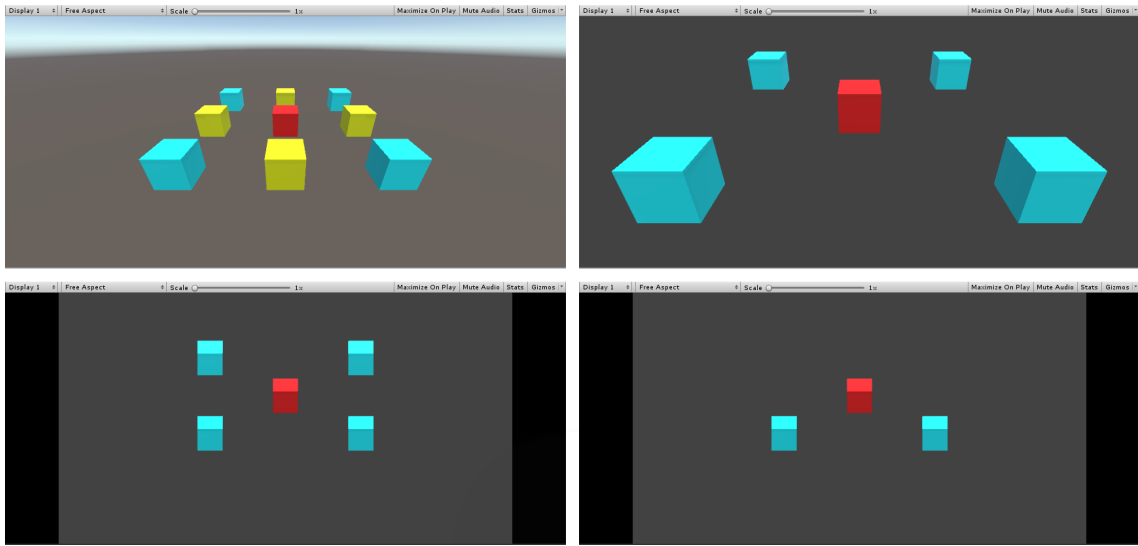


Abbildung 2.2.: Vier verschiedene Resultate beim Rendering einer Beispielszene durch verschiedene Kamera-Einstellungen

angezeigt. Damit daraus das zweite Bild entsteht, wird den gelben Würfeln ein separater *Layer* zugewiesen und dieser über das „Culling Mask“-Dropdown des *Camera*-Skripts deaktiviert. Der Wert des „Field of View“-Parameters wird abgesenkt, um die Objekte näher erscheinen zu lassen. Zuletzt wird über die „Clear Flags“-Auswahl festgelegt, dass nicht mehr die Skybox, sondern eine deckende Hintergrundfarbe dort gerendert werden soll, wo keine Objekte in der Szene existieren. Auf den unteren beiden Bildern ist die Projektion orthogonal statt perspektivisch, alle Würfel wirken trotz ihrer unterschiedlichen Entfernung zur Kamera gleich groß. Außerdem wurden die „Viewport Rect“-Parameter angepasst, um das Kamerabild nur noch auf 80% des Viewports anzuzeigen. Zusätzlich bewirkt hier eine Verschiebung um 10% auf der X-Achse die Zentrierung des Kamerabilds im Viewport. Im rechten unteren Bild sind nur noch drei Würfel sichtbar. Dies liegt daran, dass über die „Clipping Planes“-Einstellungen ein sehr geringer „Far“-Wert definiert wurde. Sind Flächen weiter vom Kamera-Objekt entfernt platziert, als dieser Maximalwert angibt, werden sie beim Rendern nicht mehr berücksichtigt. Mit einem korrespondierenden „Near“-Wert wäre es beispielsweise auch möglich, nur noch den roten Würfel rendern zu lassen.

2.3.4. Raycasting

Raycasting bezeichnet im Kontext der *Unity Engine* eine Methode, in der Szene befindliche *Collider* zu detektieren und Eigenschaften des korrespondierenden Objekts abzufragen. Dazu wird ein Strahl (*Ray*) einer bestimmten Länge von einem bestimmten Punkt aus in eine bestimmte Richtung geworfen. Trifft er auf einen *Collider*, werden detaillierte Informationen über diesen zurückgegeben, auf deren Basis sich Parameter verändern oder Funktionen aufrufen lassen. [Uni17a, Physics.Raycast]

2.4. XML

Bei XML (kurz für Extensible Markup Language) handelt es sich um eine Beschreibungssprache für Daten. Ein wesentlicher Vorteil von XML-Dokumenten besteht darin, dass es sich um Texte handelt, die sich mit einem normalen Editor erstellen und bearbeiten lassen. Die enthaltenen Daten können von unterschiedlichen Programmen aufgabenspezifisch dargestellt und bearbeitet werden.

2.4.1. Aufbau eines XML-Dokuments

XML-Dateien sind hierarchisch aufgebaut. An erster Stelle steht immer das sog. Dokumentelement, welches den Anfang der Daten kennzeichnet. Es enthält beliebig viele andere Elemente, denen sich wiederum Elemente unterordnen lassen - so entsteht eine Baumstruktur. Damit ein XML-Dokument als „wohlgeformt“ bezeichnet werden kann, muss es aus korrekten XML-Ausdrücken bestehen und darf keine Fehler enthalten. Ein einzelnes Element setzt sich aus zwei Markierungen, die den Anfang und das Ende der Daten beschreiben, sowie den Daten selbst zusammen. Die Art der Daten ist hier zunächst irrelevant, bei ihnen kann es sich um Text, numerische Werte, andere Elemente oder auch gemischten Inhalt (sowohl Elemente als auch Zeichendaten) handeln.

Damit eine Anwendung die in einem XML-Dokument beschriebenen Daten verarbeiten kann, reicht dessen Wohlgeformtheit nicht aus - es werden zusätzliche Informationen benötigt. Dazu zählen Name und Aufbau der Elemente sowie die Definition der Datenstruktur. Nur, wenn ein XML-Dokument frei von Syntaxfehlern ist und zusätzlich einem vorgegebenen Regelwerk gehorcht, erfüllt es alle Voraussetzungen

<div> <div>1</div> <div>H</div> <div>Wasserstoff</div> <div>1,0097</div> <div>1</div> <div>2.1</div> </div>	<pre> <atom name="Wasserstoff" kuerzel="H"> <ordnungszahl>1</ordnungszahl> <atomgewicht>1</atomgewicht> <schalen>1</schalen> <valenzelektronen>1</valenzelektronen> <elektronegativitaet>2,1</elektronegativitaet> <protonen>1</protonen> <elektronen>1</elektronen> <neutronen>0</neutronen> <serie>Nichtmetalle</serie> </atom> </pre>
---	--

Abbildung 2.3.: Notation von Wasserstoff im Periodensystem und die daraus abgeleitete Beschreibung von Wasserstoff als XML-Element. Die Farbgebung eines Kästchens im Periodensystem kennzeichnet für gewöhnlich die Serie des Elements (hier: hellgrün entspricht Nichtmetallen).

eines sogenannten gültigen Dokuments. Hier gibt es zwei Regelwerke, die den Aufbau von Dokumenten (auch: Dokumentklassen) mit allen erlaubten Varianten definieren können: DTD (Document Type Definition) und XML-Schema. Ein XML-Schema ist XML-konform, d.h. es handelt sich um ein reines XML-Dokument; gegenüber einer DTD bietet es mehr Möglichkeiten, Regeln festzulegen. So können unter anderem Datentypen für Element- und Attributinhalt wie auch genaue Multiplizitäten angegeben werden, was es zu einem insgesamt flexibleren Werkzeug macht [Erl03, S.140].

2.4.2. Anwendungsmöglichkeiten in Unity

Die XML-Syntax eignet sich hervorragend zur Anlegung von Datenbanken. Dies kann sofort auf ein Anwendungsbeispiel übertragen werden: Das Periodensystem der Elemente lässt sich ohne weiteres in eine XML-Datenbank konvertieren - und das im wörtlichen Sinn: Jedes chemische Element wird durch ein XML-Element beschrieben. Letzteres enthält wiederum Elemente, die die Eigenschaften des Atoms wiedergeben. Abbildung 2.3 zeigt die direkte Gegenüberstellung.

Die Generierung von 3D-Modellen auf Basis einer XML-Datenbank bietet sich also an. Allerdings dürfte sie bei Molekül-Modellen äußerst schwierig ausfallen, da Faktoren wie die Bindungswinkel und Ladungsverteilungen hier Berücksichtigung

finden müssten. Aufgrund dessen wird dieser Ansatz in AtomCAD zunächst mit Atom-Modellen erprobt. Im Fall einer erfolgreichen Umsetzung würde sich die Notwendigkeit erübrigen, bei einer Erweiterung des Programms weitere Komponenten in einem 3D-Programm modellieren zu müssen. Nutzer wie auch Entwickler könnten noch nicht enthaltene chemische Elemente einfach in der jeweiligen XML-Datei ergänzen.

Abgesehen von der dynamischen Generierung der Reaktionskomponenten kommt XML auch für die Speicher- und Ladefunktionen von AtomCAD infrage.

3. Anforderungen und Spezifikation

Um die Visualisierung der Beispielreaktion realisieren zu können und gleichzeitig eine erweiterbare Basis zu schaffen, sollen in diesem Kapitel die für den Prototyp des Programms notwendigen Elemente sowie Funktionen festgelegt und deren Umsetzung geplant werden. Ergänzend dazu beschäftigen sich die letzten beiden Abschnitte des Kapitels mit Anforderungen an die Nutzeroberfläche unter dem Aspekt der Usability sowie den Alleinstellungsmerkmalen des Programms.

3.1. Aufbau der grafischen Benutzerschnittstelle

Eine detaillierte Festlegung und Spezifizierung der einzelnen Funktionen von AtomCAD wird erst während der Implementierung möglich sein. Anders verhält es sich dagegen mit dem groben Aufbau des Graphical User Interfaces. Die Beschreibung und Einordnung der für die spätere Gliederung der Funktionen erforderlichen GUI-Elemente sollen aus diesem Grund an erster Stelle der Anforderungsanalyse stehen.

3.1.1. Hauptmenü

Als Ausgangspunkt der Navigation in AtomCAD wird zunächst ein Hauptmenü benötigt. Die in diesem enthaltene Funktionen lassen sich bereits definieren. Zum einen muss der Nutzer neue Reaktionen anlegen bzw. von ihm angelegte oder schon in der Anwendung vorgefertigte Reaktionen laden können. Um AtomCAD für verschiedene Bildschirmgrößen zu optimieren, empfiehlt sich außerdem, in einem verknüpften Untermenü eine Reihe von geläufigen Auflösungen anzubieten.

3.1.2. Auswahlmenü

Will der Nutzer bereits gespeicherte Reaktionen laden, wird ein Menü benötigt, über das sich Reaktionen abrufen, umbenennen sowie entfernen lassen. Um zu verhindern, dass der Nutzer aus Versehen vorgefertigte Reaktionen löscht, sollte eine separate Auswahlfunktion in einer anderen Szene angelegt werden. Diese wird im Folgenden als „Schaukasten“ bezeichnet. Die Inklusion eines solchen Menüs empfiehlt sich vor allem, um der Überschreibung von Muster-Reaktionen entgegenzuwirken. Der Schaukasten könnte aber auch dazu dienen, Hintergrundinformationen über den jeweiligen Reaktionsmechanismus zu vermitteln. Die Animation des Reaktionsablaufs ließe sich so direkt in den Zusammenhang eines Seminars, einer Übung oder auch einer Vorlesung setzen.

3.1.3. Szene-GUI

Innerhalb der geladenen Szene muss dem Nutzer ein umfangreiches GUI zur Verfügung stehen, über das er die benötigten Komponenten in der Reaktionsumgebung definieren, generieren sowie editieren kann. Neben dem Zugriff auf die Speicherfunktionen sollten in diesem Szene-GUI außerdem Funktionen implementiert werden, die die Eingabe verschiedener Reaktionsparameter regulieren.

Von den in 2.3.1 beschriebenen *Render Modes* eignet sich „Screen Space - Overlay“ für das *Canvas*-Objekt des Szene-GUIs am besten, da die grafische Nutzeroberfläche von AtomCAD später jederzeit in der Szene fixiert, sicht- und bedienbar sein soll. Zwar würde „Screen Space - Camera“ die Darstellung der UI-Elemente mit räumlicher Tiefe erlauben; es ist allerdings anzunehmen, dass die Kamera oft bewegt wird. Das Risiko, dass während des Renderings Fehler entstehen, wenn diese in geringer Entfernung zu Objekten platziert wird, ist bei dieser Einstellung vermutlich größer.

3.2. Reaktionsparameter

Um den korrekten Ablauf der Beispielreaktion in der Reaktionsumgebung zu gewährleisten, sind reaktionsspezifische Parameter zu berücksichtigen, deren Anpassung teilweise durch den Nutzer geschehen soll.

Wie bereits beschrieben läuft die elektrochemische CO_2 -Reduktion an einer Kathode ab. Manche Reaktionen dürfen im Umkehrschluss also ausschließlich dann stattfinden, wenn eine vorher definierte „Kontaktbedingung“ erfüllt ist. Für Reaktionen, die elektrolytisch erzwungen werden, eignet sich eine Differenzierung zwischen zwei Typen: Der Kontakt mit Kathoden (hier: negativ geladene Elektroden) sowie Anoden (hier: positiv geladene Elektroden). Im Anwendungsfall wirken die Elektroden als Katalysatoren; erst durch die angelegte Spannung wird der für die Reaktion nötige Energieaufwand aufgebracht. Die Auswahl der entsprechenden Objekte in der späteren Szene sollte dem Nutzer überlassen werden. Eine solche Kontaktbedingung ließe sich in *Unity* durch eine *Collider*-Abfrage implementieren.

Ebenfalls zu beachten ist der pH-Wert des Elektrolyts, in dem die Reaktion abläuft. Wie auf Abbildung 2.1 zu sehen ist, ist einer der möglichen Reaktionsverläufe abhängig vom pH-Wert. Hier empfiehlt sich der Einbau eines *Input Fields*, in welches der Nutzer den gewünschten Wert eintragen kann.

3.3. Grafische Illustrierung des Reaktionsablaufs

Für die ansprechende Visualisierung der Reaktion muss ein geeignetes Modell gefunden werden. Dies beinhaltet unter anderem die Darstellung von Atomen und Molekülen - wie detailliert die chemischen Komponenten abgebildet werden, nimmt später großen Einfluss auf die Verständlichkeit der Animation. Auch zu berücksichtigen ist, dass die Reaktionsabläufe nicht zu abstrahiert visualisiert werden; gleichzeitig besteht der Nachteil einer vergleichsweise akkuraten Darstellung neben der schwierigeren Implementierung darin, dass das Gesamtbild darunter leidet. Sind die einzelnen Komponenten zu detailliert, ist es möglich, dass sie die Aufmerksamkeit des Betrachters von den relevanten Reaktionsmechanismen ablenken, statt zu deren Verständnis beizutragen. Trotzdem darf die Visualisierung nicht so abstrakt ausfallen, dass wichtige Reaktionsphasen oder Zwischenschritte unzureichend nachvollzogen werden können.

3.3.1. Darstellung chemischer Komponenten

Visualisierungen chemischer Verbindungen im dreidimensionalen Bereich beschränken sich häufig auf ein Äquivalent der Valenzstrichformel. Hierbei schenken die

3D-Modelle freien Elektronenpaaren im Regelfall keine Beachtung, sondern stellen ausschließlich die Bindungen zwischen den Atomen sowie die räumliche Anordnung der Atome im Molekül dar (s. Tabelle B.1.2).

Dieses Modell zu adaptieren, wurde für AtomCAD in Betracht gezogen, erwies sich jedoch bereits nach kurzer Zeit als problematisch. Gerade bei komplexen Reaktionen, die wie die elektrochemische CO₂-Reduktion an Elektroden bzw. katalytisch ablaufen, würde diese Art der Visualisierung schlichtweg inakkurat ausfallen oder aber mit beträchtlichem Modellierungs-Aufwand verbunden sein. Dies liegt darin begründet, dass während des Gesamtablaufs der Reaktion regelmäßig Bindungen aufgelöst sowie neu geformt werden - Bindungswinkel innerhalb eines Moleküls ändern sich, es kommt zu Ladungsverschiebungen. Da neue Moleküle in diesem Vorgang nicht in einem einzigen Schritt entstehen, sondern sich Komponenten im Gegensatz dazu nach und nach neu anordnen und bilden, reicht es nicht aus, die Reaktion nach dem Additionsprinzip zu betrachten.

Aus diesen Überlegungen lässt sich eine Visualisierungsmethode ableiten, die am ehesten vergleichbar mit dem sog. CPK-Modell [Kol65] ist. Auf den ersten Blick enthält dieses vergleichsweise wenig Informationen über das Molekül. Farbige Kugeln stellen die einzelnen Atome dar, ersichtlich sind lediglich die Größenverhältnisse sowie Bindungswinkel. Anzahl und Art der Bindungen finden dagegen keine Berücksichtigung. Dies hat gegenüber der Valenzstrichformel den Vorteil, dass es nicht zu einer falschen Abbildung von Bindungen kommen kann, setzt aber voraus, dass der Betrachter über die jeweiligen Bindungsverhältnisse im abgebildeten Molekül informiert ist.

Die Farbgebung von Atomen im CPK-Modell wurde von mehreren Programmen adaptiert und erweitert. Hier zeichnet sich besonders das Open-Source-Tool *Jmol* ab, welches fast allen Elementen des Periodensystems eine einzigartige Farbe zuweist [Jmo17]. Die *Jmol*-Farbtabelle für AtomCAD zu übernehmen, bietet sich deshalb an.

3.3.2. Visualisierung der Reaktionsmechanismen

Die Anforderungen an die Reaktionsvisualisierung werden im Folgenden zunächst auf den Beispielfall der elektrochemischen CO₂-Reduktion bezogen erläutert. Ziel ist

hierbei, eine Herangehensweise zu konzipieren, die später als einheitliche Grundlage auf beliebige Reaktionsmechanismen anwendbar ist.

Die Beispielreaktion beginnt damit, dass Kohlenstoffdioxid auf eine Kathode trifft und an dieser reduziert wird. Zunächst kann also von einem CO_2 -Molekül-Modell ausgegangen werden, welches sich im 3D-Raum bewegt. Kollidiert es mit einem Kathoden-Modell, muss sich die Übergabe von Elektronen für den Betrachter erschließen. Hierfür empfiehlt sich, dem Modell ein sog. *Text Mesh* zuzuordnen. In dieses wird zu Beginn nichts eingetragen, damit es für den Betrachter unsichtbar ist. Im Fall einer Kollision können nun per Skript die Zuweisung der Ladung sowie deren Anzeige über das *Text Mesh* erfolgen.

Kollidiert das CO_2 -Molekül dagegen mit einem anderen Molekül, müssen gleich mehrere Abfragen erfolgen. An erster Stelle steht die Prüfung, ob sich das Molekül an einer Kathode befindet und geladen ist, denn nur dann kann im Beispielfall eine Reaktion stattfinden. Da aber nicht alle Reaktionen elektrochemisch ablaufen, bietet sich die Nutzung von boolschen Werten in einer späteren Molekül-Klasse an. Über diese ließen sich die Bedingungen definieren, unter denen das jeweilige Molekül reagiert.

Einen Spezialfall stellt die Kollision mit Protonen dar. Bis auf wenige Zwischenschritte lässt sich der Ablauf der CO_2 -Reduktion damit beschreiben, dass stetig Protonen auf die an die Kathode gebundenen Moleküle treffen, wo sie zu Wasserstoff reduziert werden. Bis stabile neue Moleküle entstehen, müssen zuvor oft erst mehrere Protonen reduziert werden; gleichzeitig bedarf das Entstehen neuer H-Teilchen mit Bindung zum Molekül einer Visualisierung.

Trifft ein Proton auf ein geladenes CO_2 -Molekül, könnten jedes Mal beide 3D-Modelle gelöscht sowie an dieser Stelle die daraus entstandene Verbindung instanziiert werden. Dies erscheint jedoch wenig sinnvoll und würde wahrscheinlich visuell ablenkend wirken. Ein möglicher Kompromiss besteht darin, lediglich die Protonen bei Kollisionen mit negativ geladenen CO_2 -Molekülen zu löschen und dafür an einem vordefinierten Punkt am Molekül-Modell ein Wasserstoff-Modell zu instanziiieren. So müsste das Molekül erst gelöscht werden, wenn sich die Eigenschaften der Verbindung stark ändern oder aus der Reaktion mit einem Proton mehrere Verbindungen resultieren.

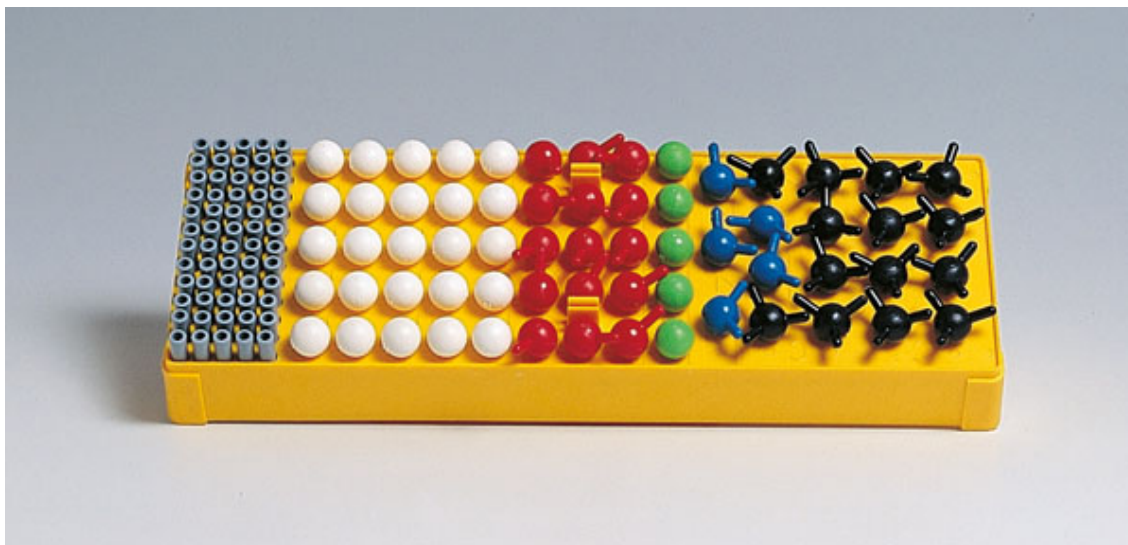


Abbildung 3.1.: Molekül-Baukasten von Leybold: Besonders gut sichtbar sind die Verbindungsstäbe an den schwarzen Kohlenstoffmodellen. [Aus: <https://www.ld-didactic.de/phk/images/150dpi/CE18474.jpg>, 02.11.2017]

Bei der Implementierung des Prototypen wird wie in 2.4.2 beschrieben getestet, inwiefern sich Atome mit Informationen aus einer XML-Datei generieren lassen. Fällt die Umsetzung erfolgreich aus, könnte ein modulares Modell für Moleküle in Betracht gezogen werden, welches einem ähnlichen Prinzip folgt wie physische Molekülbaukästen (s. Abbildung 3.1): Atome könnten mit „Andock-Punkten“ generiert werden, deren Anzahl der Anzahl an Elektronen entspricht, die dem Atom zum Erreichen der Edelgaskonfiguration fehlt. Im Idealfall müsste keine Komponente mehr gelöscht, sondern nur noch „entkoppelt“ und neu angeordnet werden - dies würde eine wesentlich genauere Reaktionsvisualisierung erlauben. Jedoch wäre die Visualisierung noch immer nicht akkurat, solange sich diese „Andock-Punkte“, also in etwa *Transform*-Objekte, an festen Positionen am Atom-Modell befinden würden. Als Beispiel ist hier Kohlenstoff zu nennen, welches verschiedene mögliche Hybridisierungen hat. Je nach Hybridisierung unterscheiden sich die Positionen der Hybridorbitale und damit die Bindungswinkel im Molekül.

3.3.3. Kamerabewegung

Neben der ansprechenden Darstellung der chemischen Komponenten und kohärenten Visualisierung der Reaktion spielt auch der Winkel eine Rolle, in dem der Nutzer die Reaktion betrachtet.

Die in Abschnitt 2.3.4 beschriebene Methode des Raycastings kann hierbei verwendet werden, um die Kamera abhängig von den Eingaben des Nutzers zu bewegen. Per Klick lässt sich so beispielsweise festlegen, auf welche Komponente diese gerichtet werden soll. Die Rotation der Kamera um ausgewählte Komponenten wäre ebenfalls eine Option.

Der besseren Veranschaulichung der Reaktion wegen empfiehlt sich darüber hinaus die Implementierung von mehr als einer Kamera. Wie und wann zwischen deren Perspektiven gewechselt wird, sollte dem Nutzer überlassen sein. Eine beliebige Positionierung kombiniert mit der Möglichkeit, jederzeit andere Kameras zu aktivieren, würde die individuelle Anpassung des Blickwinkels gewährleisten.

3.4. Assets

Dieser Abschnitt soll dazu dienen, eine Übersicht der benötigten Assets zu schaffen. Dazu zählen die 3D-Modelle der Moleküle, diesen zuzuweisende *Materials* bzw. *Physic Materials* sowie UI-Grafiken. C#-Skripts und XML-Datenbanken finden hier zunächst keine Berücksichtigung.

3.4.1. 3D-Modelle

Zentraler Bestandteil der Reaktionsvisualisierung sind die 3D-Modelle der Reaktionskomponenten. Da Moleküle wie beschrieben mit Kugeln dargestellt werden, fallen Komplexität und Modellieraufwand der Komponenten vergleichsweise gering aus. Für die Erstellung wird Autodesk Maya 2016 verwendet - weil die Modelle aus unveränderten Kugel-Grundformen bestehen, spricht jedoch nichts gegen eine Umsetzung in beliebigen 3D-Programmen.

Trotz der simplen 3D-Struktur der Molekül-Modelle müssen mehrere Umstände berücksichtigt werden. Zum einen ist eine einheitliche Darstellung nötig. Dies schließt

ein, dass es zwischen den Kugeln fest definierte Abstände gibt, die es einzuhalten gilt. Diese Abstände fallen je nach Art der Verbindung unterschiedlich groß aus. Ein weiterer wichtiger Aspekt ist die Schaffung eines Maßstabs, der die Atomradien der einzelnen Elemente korrekt abbildet. Als Referenz empfiehlt sich hier eine Kugel, deren Duplikate auf die gewünschte Größe skaliert werden.

Damit der Nutzer Atome und Moleküle in der Szene identifizieren kann, spielt außer der Form der 3D-Modelle auch deren Farbe eine Rolle. Hier wird wie in 3.3.1 ausgeführt das *Jmol*-Farbmodell adaptiert; die jeweiligen *Materials* werden zentral als Prefab definiert und später über ihren Namen referenziert, sodass mit der Instanziierung eines Modells kein neues Material angelegt werden muss.

Eine Besonderheit von AtomCAD ist, dass Protonen ebenfalls als farbige Kugeln dargestellt werden. Deren Radius fällt etwas kleiner aus als der eines Wasserstoffmodells - maßstabgerecht ist dies zwar nicht, schließt aber Verwechslungen aus und gewährleistet gute Sichtbarkeit. Darüber hinaus ist es unabdingbar, bei der Beispielreaktion zwischen Protonen und Wasserstoffatomen zu unterscheiden.

3.4.2. Materials

Unity bietet die Möglichkeit, 3D-Modelle mit ihren *Materials* (auch als Textur bezeichnet) zu importieren. Diese Vorgehensweise hat jedoch den Nachteil, dass mit jedem importierten Objekt separate *Materials* angelegt werden. Einem Wasser-Molekül und einem Kohlenstoffdioxid-Molekül könnte beispielsweise ein *Material* mit roter Farbe zugewiesen sein, welches das Sauerstoff-Atom in der Verbindung kennzeichnet. Beim Import würde für jedes Objekt ein einzelnes rotes *Material* angelegt werden.

Die Alternative dazu besteht in der zentralen Definition von *Materials* in einem separaten Projekt-Ordner innerhalb von *Unity*. Für jedes chemische Element wird nach der *Jmol*-Farbtabelle ein *Material* vorgefertigt. Deren Zuweisung an die 3D-Modelle gestaltet sich zwar aufwändiger als der einfache Import, gewährleistet aber, dass es nur ein *Material* für jeden Atomtyp gibt und die Darstellung somit auf jeden Fall einheitlich bleibt.

3.4.3. Physic Materials

Physic Materials beschreiben die physikalischen Eigenschaften eines Objekt-*Colliders* – dazu zählen statische Reibung, dynamische Reibung und Elastizität [Uni17b, Physic Material]. Um eine kontinuierliche Bewegung der Komponenten in der Reaktionsumgebung zu gewährleisten, könnte ihnen ein *Physic Material* zugewiesen werden, welches keine Reibung aufweist. Da die Darstellung von Atomen als Kugel extrem abstrahiert ist, bedarf es während der Implementierung von AtomCAD einer Analyse von verschiedenen Parameter-Konfigurationen, um geeignete Attribute für die *Physic Materials* festzulegen.

3.4.4. Grafiken

In *Unity* enthaltene, vorgefertigte UI-Komponenten gestatten die Personalisierung der Nutzeroberfläche, indem sie Möglichkeiten bieten, ihnen eigene Grafiken zuzuweisen. Schaltflächen können so beispielsweise deskriptive Icons hinzugefügt werden, die dem Nutzer auf einen Blick deren Funktionalität vermitteln. Obwohl es sich hierbei um rein kosmetische Maßnahmen handelt, sollte berücksichtigt werden, dass die Navigation der Nutzeroberfläche durch ein passendes grafisches Konzept erheblich erleichtert wird. Somit zählt auch das Grafikdesign zur Usability des Programms.

3.5. Usability

AtomCADs primäres Ziel ist, Wissen anschaulich zu vermitteln. Es soll im Bildungsbereich zum Einsatz kommen und von Lehrern und Dozenten verwendet werden. Damit die Nutzung von AtomCAD einen Mehrwert darstellt, muss zum Beispiel sichergestellt sein, dass die Einarbeitung ohne großen Zeitaufwand erfolgen kann und möglichst intuitiv ist. Diese Anforderungen lassen sich unter dem Sammelbegriff der Usability zusammenfassen.

3.5.1. Definition

Usability (auch Gebrauchstauglichkeit, Nutzbarkeit oder Nutzerfreundlichkeit) wird durch die ISO-Norm 9241-11 als das Ausmaß beschrieben, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen [DIN98, S.225]. Der Begriff fasst wiederum mehrere Teilaspekte zusammen – DIN EN ISO 9241-110 definiert sieben Eigenschaften, anhand derer die Beurteilung der Usability einer Benutzerschnittstelle erfolgt (s. B.2.1).

3.5.2. Angewandte Methoden

Das GUI von AtomCAD bereits in der prototypischen Implementierungsphase detailliert nach den einzelnen Aspekten der Usability aufzuschlüsseln und zu bewerten, würde den Rahmen dieser Arbeit sprengen. Dessen Nutzerfreundlichkeit ist im Fall einer Weiterentwicklung des Programms aber in jedem Fall zu berücksichtigen. Methoden des Usability Engineerings sowie der Prüfung von Usability sind vielfach dokumentiert [Nie93, NM94].

Trotzdem sollen einige Facetten des Usability Engineerings gezielt in den Designprozess des GUIs einfließen. Ein wichtiges Beispiel ist hierbei die Anwendung der sog. Gestaltprinzipien bei der Anordnung und Farbgebung von Icons und Menüs. Sie beschreiben, wie der Mensch visuelle Eindrücke zu Objekten organisiert [Fri17]. Eine Auflistung einiger Gestaltprinzipien mit praktischen Beispielen ist unter B.1.1 beigelegt.

3.6. Alleinstellungsmerkmale

Ausgehend von den in Abschnitt 2.1 analysierten vergleichbaren Anwendungen sowie den in diesem Kapitel identifizierten Anforderungen an AtomCAD kristallisieren sich zwei hauptsächliche Alleinstellungsmerkmale des Programms heraus.

Zum einen handelt es sich zwar um ein Visualisierungs- aber kein Zeichenwerkzeug. Eine Konvertierung von chemischen Formeln vom zweidimensionalen in den dreidimensionalen Raum findet nicht statt.

Zum anderen kann die Reaktionsanimation an sich schon als Alleinstellungsmerkmal angesehen werden. Bis auf eine Ausnahme erlaubt keine der zahlreichen vergleichbaren Anwendungen dem Nutzer, chemische Reaktionen selbstständig im 3D-Raum aufzubauen. Einzig in *CellUnity* findet eine Reaktionsanimation statt. Bei den Molekülen, die der Nutzer in die Anwendung importieren kann, handelt es sich jedoch um Proteine. Der für deren Visualisierung relevante Maßstab erfordert eine Abstraktionsebene, die verhindert, dass die genauen Reaktionsmechanismen auf atomarer Ebene nachvollzogen werden können.

4. Implementierung

Dieses Kapitel widmet sich der Implementierung des AtomCAD-Prototypen. Es beschreibt Aufbau und Funktionen der einzelnen Menüs und die Gestaltung der Reaktionsumgebung. Die implementierten C#-Skripte sowie ein umfangreiches Klassendiagramm sind auf der beiliegenden CD zu finden, außerdem ist im Anhang unter B.3 eine Auflistung enthalten, die eine kurze Beschreibung der Klassen beinhaltet.

4.1. Menüs

Neben der Szene, in der später die Reaktion stattfindet, wurden drei Menüs angelegt, die der Navigation des Nutzers dienen und im Fall einer Weiterentwicklung auch Zugriff auf verschiedene Einstellungen erlauben könnten.

4.1.1. Startmenü

Das Startmenü enthält fünf Schaltflächen: „Neue Reaktion“, „Reaktion laden“, „Schaukasten“, „Optionen“ sowie „Beenden“. Da noch keine Speicher- und Ladefunktionen implementiert wurden, erfolgt der Abruf der Reaktionsumgebung bisher mit einem einfachen Wechsel der Szene. Wird die „Optionen“-Schaltfläche angewählt, schwenkt die Kamera auf ein separates Canvas, in dem später entsprechende UI-Elemente Platz finden können.

4.1.2. Auswahlmenüs

Auf der Grundlage der in 3.1.2 ausgeführten Überlegungen gibt es zwei verschiedene Auswahlmenüs, die sich bisher nur in einem Detail voneinander unterscheiden: Die Schaukasten-Auswahl stellt keine Schaltfläche zur Verfügung, über den später

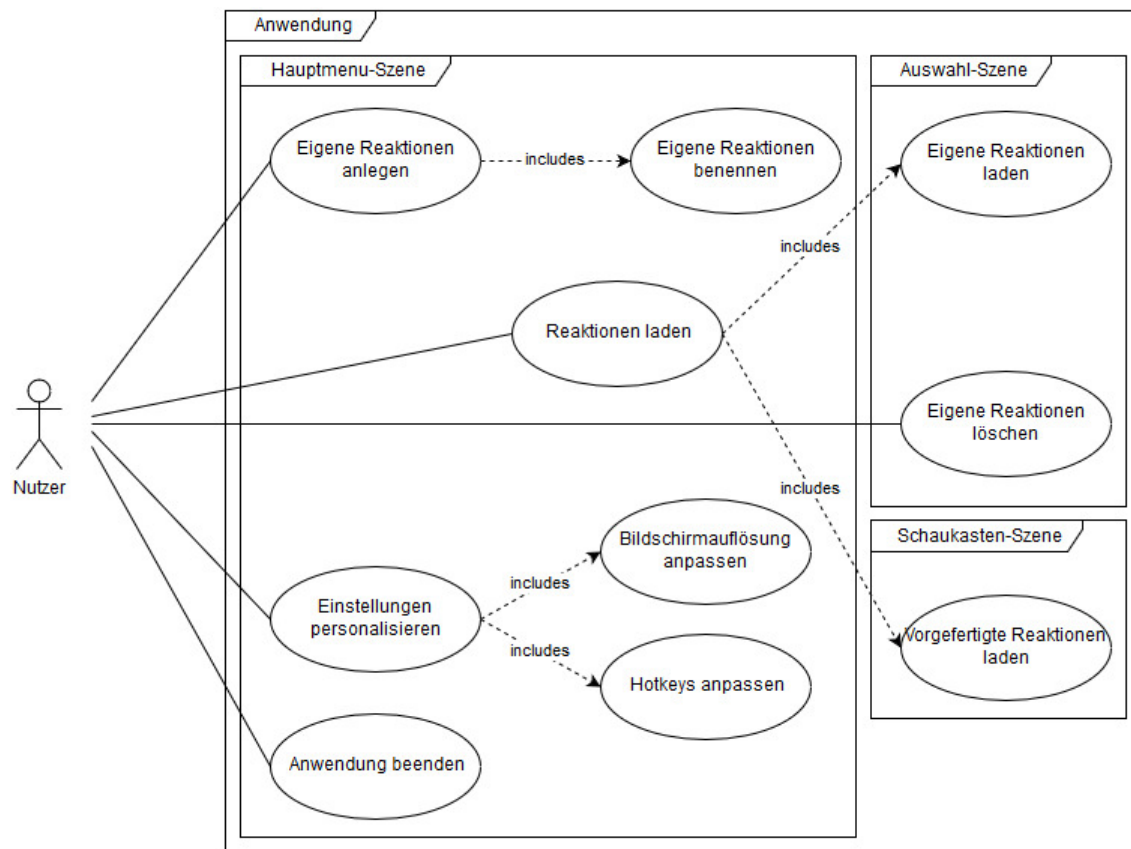


Abbildung 4.1.: Use-Case-Diagramm der Haupt- und Auswahlmenü-Struktur

eine vorgefertigte Reaktion gelöscht werden könnte. Abbildung 4.1 beschreibt die geplanten Funktionen der einzelnen Menüs sowie deren Aufteilung in verschiedene Szenen.

4.2. Aufbau des Szene-GUIs

Im Folgenden wird der Aufbau des Haupt-GUIs beschrieben, mit dem der Nutzer in einer von ihm angelegten Szene interagiert. Hierbei werden sowohl Funktionen berücksichtigt, die bereits implementiert sind als auch solche, die bei einer Weiterentwicklung von AtomCAD infrage kämen. Für letztere wurden im GUI Platzhalter eingebaut, um eine Funktionalitätserweiterung des Programms nicht dadurch zu erschweren, wichtige Komponenten in separaten Fenstern unterbringen zu müssen.

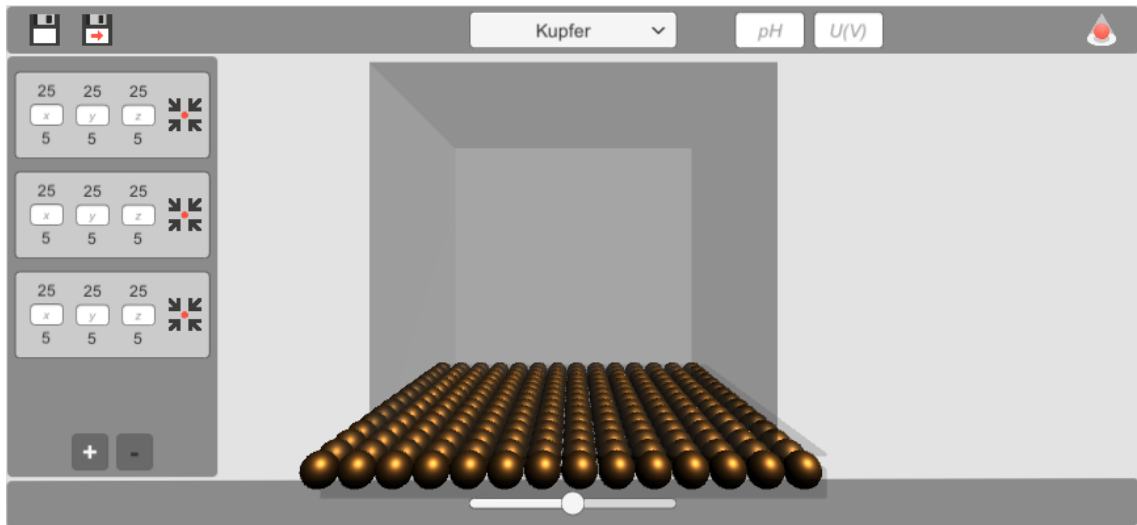


Abbildung 4.2.: Das Szene-GUI. Oben die Menüleiste, links der Spawn-Panel-Container mit drei Spawn-Panels, unten die Zeitleiste.

4.2.1. Menüleiste

Die Menüleiste enthält verschiedene Funktionen, die dem Speichern von Reaktionen sowie dem Organisieren der Nutzerschnittstelle dienen. Es gibt zwei verschiedene Speicher-Schaltflächen. Die erste soll später nur die jeweiligen Daten abspeichern. Mit der zweiten könnte der Nutzer die Reaktion unter einem anderen Namen sichern.

Ebenfalls in der Menüleiste aufzufinden sind die beiden Anzeige-Buttons, die am rechten Rand platziert sind und von denen immer nur einer eingeblendet ist. Sie erlauben die visuelle Anpassung der Benutzerschnittstelle. Bei ihrer Aktivierung bzw. Deaktivierung erfolgt das Ein- und Ausblenden der Menüleiste, des Spawn-Menüs sowie der Zeitleiste, um einen besseren Überblick der Szene zu bieten.

Neben diesen Speicher- und Organisationsfunktionen können in der Menüleiste verschiedene Einstellungen an Reaktionsparametern vorgenommen werden. Zum einen befindet sich hier das Elektroden-Dropdown, welches dem Nutzer die Auswahl eines Elektrodenmaterials erlaubt. Zum anderen kann der Nutzer hier sowohl den pH-Wert des Elektrolyten als auch die angelegte Spannung über *Input Fields* definieren.

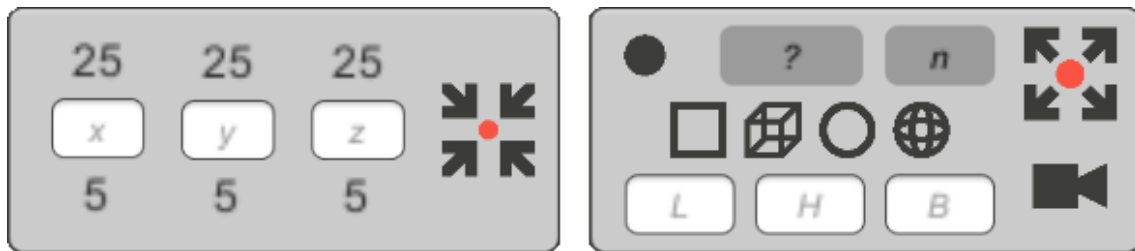


Abbildung 4.3.: Spawn-Panel, links mit Spawn-Punkt-GUI, rechts mit Spawn-GUI

4.2.2. Zeitleiste

Animationsprogramme arbeiten für gewöhnlich mit Zeitleisten. Auf diesen setzt der Nutzer Keyframes, die die Eigenschaften des animierten Objektes am gewünschten Zeitpunkt klar definieren. Dazu gehören in etwa seine Position, Rotation, Größe, Deckkraft oder Farbe. Damit eine Animation flüssig abläuft, spielt die Platzierung von Keyframes eine große Rolle. Da AtomCAD keine Grundkenntnisse auf dem Gebiet der Animation voraussetzen kann, dient die Zeitleiste ausschließlich dazu, den Reaktionsablauf zu stoppen, verlangsamen oder beschleunigen. Zu diesem Zweck befindet sich in ihr ein gewöhnlicher Slider, über den die Reaktionsgeschwindigkeit angepasst werden kann.

4.2.3. Spawn-Menü

Das Spawn-Menü ist das Herzstück des Haupt-GUIs. Es lässt sich in etwa mit dem Ebenen-Bedienfeld eines Zeichenprogramms gleichsetzen und besteht beim Anlegen einer neuen Szene lediglich aus einem leeren *Panel*-Element, dem Spawn-Panel-Container. Sobald der Nutzer darin eine Schaltfläche aktiviert, wird dem Container ein Spawn-Panel hinzugefügt. Über dieses lässt sich nun definieren, wo im virtuellen Raum welche ausgewählte Komponente generiert werden soll. Neben der Festlegung der Koordinaten eines Spawn-Punktes im 3D-Raum bietet ein Spawn-Panel zusätzlich Kontrolle über zahlreiche weitere Parameter.

Wie auf Abbildung 4.3 links zu sehen ist, enthält ein Spawn-Panel nach der Instanziierung zunächst eine editierbare Zeile, die die Position des Spawn-Punktes im 3D-Raum beschreibt. Die Textfelder über und unter den jeweiligen Eingabefeldern zeigen die minimal sowie maximal möglichen Koordinatenwerte an. Diese

sind abhängig von den Koordinaten, an denen die Reaktion stattfinden kann - zu Testzwecken wurden sie auf einen Würfel zugeschnitten, dessen Seitenlängen jeweils 30 Einheiten messen und der sich an der Position (15,0,15) befindet. Hat der Nutzer seine gewünschten Koordinaten eingegeben, kann über den letzten *Button* in der Zeile ein Spawn-Punkt instanziiert werden. Gleichzeitig wechseln die im Spawn-Panel angezeigten UI-Elemente.

Das fortan angezeigte Panel-Interface dient der Spezifikation des Spawn-Vorgangs an sich. Es besteht aus drei Zeilen. In der ersten Zeile wird durch ein *Toggle* die Art der zu instanziiierenden Komponente ausgewählt. Komponenten unterscheiden sich logisch jeweils in der Art, wie sie später erzeugt werden. Bei Atomen wird das im Index-Eingabefeld referenzierte Element aus einer XML-Datei geladen. Mit den darin beschriebenen Eigenschaften wird dann ein vorgefertigtes Objekt überschrieben. Moleküle werden dagegen direkt aus dem Ressourcen-Ordner innerhalb des Projekts geladen. Diese Unterscheidung wurde gewählt, da es sich für die Umsetzung des Prototypen als zu aufwendig herausstellte, Moleküle in XML zu beschreiben. Je nach ausgewählter Komponente bestimmt der Nutzer das zu instanziiierende Objekt durch die Angabe der Ordnungszahl bei Atomen oder der Summenformel bei Molekülen. Darüber hinaus wird in derselben Zeile festgelegt, wie viele Instanzen der Komponente erzeugt werden sollen.

In der zweiten Zeile kann der Nutzer zwischen vier Verteiler-Toggles wählen, die die spätere Spawn-Position der Instanzen im Raum kontrollieren. Wird keines davon aktiviert, wird jede Instanz an der Position des im ersten Schritt definierten Spawn-Punktes generiert. Bei einer nahezu zeitgleichen Instanziierung treten hier aufgrund der *Collider*-Komponenten möglicherweise unerwünschte physikalische Verhaltensweisen bei den Objekten auf. Um dem Abhilfe zu schaffen, dient der Spawn-Punkt, sofern ein Verteiler-Toggle ausgewählt ist, nur als Zentrums-Referenz für die jeweiligen Verteiler-Skripts. Diese generieren die Komponenten zufällig innerhalb eines Kreises, einer Kugel, eines Quadrats oder eines Würfels. Hier sind in der dritten Zeile als zusätzliche Parameter noch Durchmesser bzw. Seitenlängen anzugeben, die den gewünschten, maximal möglichen Abstand vom Spawn-Punkt definieren. Als Ergänzung dazu wären Verteiler denkbar, die Objekte gleichmäßig statt auf der Basis von Zufallswerten im Raum generieren. Mit diesen könnte der Nutzer beispielsweise Gitterstrukturen erstellen.

Am rechten Rand des Spawn-Panels ist zum einen der Spawn-Button zu finden, der den Prozess der Instanziierung unter der Berücksichtigung aller eingegebener Parameter startet. Zum anderen befindet sich darunter das Kamera-Toggle. Jedes Spawn-Panel verfügt über ein verknüpft Kamera-Objekt, das durch eine *Toggle Group* verwaltet wird. Dies erlaubt dem Nutzer, zwischen verschiedenen Kameras zu wechseln. Die zusammen mit einem Spawn-Panel generierte Kamera wird gelöscht, sobald der Nutzer dieses entfernt. Abbildung 4.4 bietet eine Übersicht der bereits implementierten sowie geplanten Funktionen.

4.2.4. Baukasten-Menü

Das Baukasten-Menü wird zunächst nur als Platzhalter in das bestehende GUI eingegliedert und ist auf Abbildung 4.2 nicht zu sehen. Es basiert auf demselben Prinzip wie sein physisches Äquivalent. In seinem oberen Teil befindet sich eine Textur. Diese wird von einer Kamera gerendert, die auf einen separaten Würfel gerichtet ist, in dem sich ein Spawn-Punkt befindet. Von dessen Position aus könnten sich später anwählbare Atome generieren lassen. Wäre ein Atom selektiert, würden Informationen zu diesem eingeblendet werden. Will der Nutzer ein Atom mit einem anderen zu einem Molekül verbinden, könnte er zunächst die Dock-Punkte des Atoms auswählen, an denen das nächste Element angebunden werden soll.

4.3. Reaktionsumgebung

Dieser Abschnitt widmet sich den Komponenten, die von Anfang an in der Szene existieren, aber nicht Teil des GUIs sind. Dazu gehören sowohl die räumlichen Beschränkungen der Reaktionsumgebung als auch die Objekte und Parameter, die beeinflussen, wie die Szene gerendert wird.

4.3.1. Aufbau der Testreaktion

Wie bereits kurz in 4.2.3 erwähnt, findet die Testreaktion innerhalb eines Würfels statt, um die Übersichtlichkeit der Reaktionsumgebung zu gewährleisten. Hierbei



Abbildung 4.4.: Use-Case-Diagramm des Haupt-GUIs (Rote Kennzeichnung: Nicht in dieser Arbeit implementierte Use Cases)

handelt es sich nicht um ein einziges Objekt, sondern sechs verschiedene, deren Eigenschaften einzeln anpassbar sind.

Bevor der Game-Modus aktiviert wird, besteht die Kathode am Boden des Würfels lediglich aus einem Quader. Diesem sind mehrere *Transform*-Objekte untergeordnet. Im Game-Modus dienen sie als Orientierungs- und Begrenzungspunkte für die automatische Generierung einer Metallgitterstruktur. Die Größe der einzelnen Atome sowie deren Farbe werden aus einer XML-Datei gelesen. Ein Skript simuliert die Anziehungskraft, die von der Kathode aus auf Protonen wirkt.

Die Seitenwände des Würfels fungieren dagegen lediglich als Beschränkung, weswegen sie außer einem *Collider* keine weiteren Komponenten enthalten. Sowohl die Kollisionserkennungs-Einstellungen der Seitenwände-*Collider* als auch die der Molekül- bzw. Protonen-*Collider* werden auf „Continuous Dynamic“ gesetzt. Dies verringert die Wahrscheinlichkeit, dass die Engine Kollisionen, die bei hoher Geschwindigkeit stattfinden, nicht registriert.

Um zu verhindern, dass sich der Würfel nach und nach mit den Reaktionsprodukten füllt und diese den weiteren Reaktionsverlauf stören sowie dessen Visualisierung beeinträchtigen, empfiehlt es sich, das Entweichen vordefinierter Komponenten zu erlauben. Der Fläche, die den Deckel des Würfels bildet, wird aufgrund dessen ein Skript zugewiesen, welches ihr Verhalten als eine Art semipermeable Membran beschreibt.

4.3.2. Rendering

Bei einer schlechten Visualisierung trägt der korrekte Ablauf der Beispielreaktion allein nicht zum Verständniskern des Betrachters bei. Neben der in 3.3 ausformulierten Erwägungen, die sich auf die Darstellung chemischer Komponenten und Visualisierung der Reaktionsmechanismen bezogen, spielt es eine ebenso große Rolle, wie diese in der Reaktionsumgebung präsentiert werden.

Hierzu zählt zum einen die passende Ausleuchtung der Szene. Die Lichtfarbe in *Unity* ist standardmäßig gelb, um Sonnenlicht zu simulieren. Um eine neutrale Belichtung zu erzielen und die Farben der *Materials* nicht zu verfälschen, ist ein Wechsel auf weißes Licht vonnöten. Auch die Wahl des Licht-Typs hat später große Auswirkungen auf das Rendering. Hier wird zunächst ein Licht vom Typ *Directional Light*

platziert, welches die Szene von der Seite beleuchtet. Während der Evaluation muss kontrolliert werden, ob dieses eine geeignete Wirkung entfaltet. Fraglich ist auch, ob ein aktivierter Schattenwurf von Objekten sich bei vielen Instanzen ablenkend auswirkt.

Damit sich der Nutzer voll und ganz auf den Reaktionsmechanismus konzentrieren kann, wird auf eine Skybox verzichtet. Stattdessen rendern die Kameras dort, wo keine Objekte in der Szene existieren, eine hellgraue Hintergrundfarbe. Die Reaktionsumgebung wirkt auf diese Weise schlicht und minimalistisch. Die Eingaben des Nutzers und die von ihm instanziierten Komponenten rücken stärker in den Vordergrund.

5. Evaluation

Dieses Kapitel dient der Evaluation der in Kapitel 4 beschriebenen Programmfunktionalität. Zunächst wird der Ablauf der Reaktion observiert. Dies beinhaltet eine Analyse des Reaktionsverlaufs unter verschiedenen Bedingungen. Auf dieser Basis sollen anschließend die identifizierten Probleme dokumentiert werden.

5.1. Reaktionsverlauf

Im Folgenden wird getestet, ob sich durch die Bedienung des implementierten User Interfaces die Voraussetzungen für die Visualisierung der elektrochemischen CO₂-Reduktion schaffen lassen. Ausgehend davon wird anhand einiger Szenarien analysiert, wie sich Veränderungen an einzelnen Parametern auf den Reaktionsverlauf auswirken. Die Systeminformationen des PCs, mit dem die Tests durchgeführt werden, sind Tabelle 5.1 zu entnehmen.

5.1.1. Funktionskontrolle

Wird der Play-Modus gestartet, füllt der Elektroden-Manager zunächst das Elektroden-Dropdown mit den Namen der in der zugehörigen XML-Datei definierten Elektroden. Das erste Elektroden-Element in der Liste ist Aluminium, das Skript überlädt das *Material* des Kathoden-Gitters mit dem für Aluminium festgelegten, vorgefertigten *Material*. Wählt der Nutzer einen anderen Typen aus dem Dropdown, ist die erneute Überladung des *Materials* und die damit einhergehende Farbveränderung zu beobachten (siehe A.1.1).

Wird der kleine „+“-Button im Spawn-Menü betätigt, kommt es zu der Instanziierung eines Spawn-Panels sowie einer zusätzlichen Kamera. Wählt der Nutzer

Systemkomponente	Spezifizierung
Betriebssystem	Windows 10 Pro 64-Bit-Version (10.0, Build 16299)
Prozessor	AMD FX(tm)-9370 Eight-Core (8 CPUs), 4.40 GHz
Speicher	16384 MB RAM
DirectX-Version	DirectX 12
Grafikkarte	NVIDIA GeForce GTX 780
Speichergroße	3 GB GDDR5
Anzeigemodus	1600 x 900 (32 bit) (60 Hz)
Unity-Version	2017.2.0f3 (64-bit)

Tabelle 5.1.: Systeminformationen

ein Spawn-Panel aus und klickt auf den „-“-Button im Spawn-Menü, werden beide Komponenten gelöscht.

Gibt der Nutzer in einem Spawn-Panel Koordinatenwerte ein, die zu groß oder klein sind, werden nach der Validierung der Eingabe die höchsten oder niedrigsten möglichen Werte im korrespondierenden *Input Field* angezeigt. Die gültigen Werte kann der Nutzer darüber hinaus immer den jeweiligen Textfeldern über und unter den Eingabefeldern entnehmen. Betätigt der Nutzer den Spawn-Punkt-Button, wird an den eingegebenen Koordinaten ein Spawn-Punkt erzeugt. Um hier visuelles Feedback zu geben, besteht dieses *GameObject* nicht nur aus einem *Transform*, sondern besitzt auch ein *Sphere-Mesh*. Der Spawn-Punkt ist so als kleine Kugel mit grünem, halbtransparentem *Material* zu identifizieren.

Wird das Typ-Toggle im Atom-Modus gelassen und gibt der Nutzer gültige Werte für die Ordnungszahl und Anzahl der Instanzen ein, lässt sich der Spawn-Button aktivieren und die Instanziierung des gewünschten Atoms läuft ab. Material, Gewicht und Größe der Atome werden korrekt aus der Periodensystem-XML-Datei gelesen (siehe A.1.2). Wird das Typ-Toggle in den Molekül-Modus geschaltet und gibt der Nutzer den Namen einer bereits im Prefab-Ordner vordefinierten Verbindung ein, erfolgt die Instanziierung des gewünschten Moleküls. Die Verteiler funktionieren auf grundlegender Ebene, verfügen aber noch nicht über zusätzliche Kontrollparameter, die eine Instanziierung von Komponenten außerhalb des Testwürfels verhindern. In den späteren Testszenarien muss darauf Rücksicht genommen werden. Bei einer hohen Anzahl an Komponenten und oder kurzer Seitenlängen-Parameter sind die

geometrischen Grundformen zu erkennen, auf deren Basis die zufällige Instanziierung erfolgt (siehe A.1.3).

Aktiviert der Nutzer das Kamera-Toggle eines Spawn-Panels, wechselt die Perspektive. Befinden sich geladene Moleküle in der Szene, wird den Ladungsanzeigen die aktivierte Kamera als Referenz-Objekt zugewiesen. Auch nach mehrmaligem Kamerawechsel ist die Ausrichtung der Text-Meshes korrekt, sodass die Anzeige lesbar bleibt. Löscht der Nutzer ein Spawn-Panel, während die korrespondierende Kamera aktiviert ist, wechseln Perspektive und Ladungsanzeigen vor deren Zerstörung automatisch auf die Standard-Kamera, um Fehlermeldungen vorzubeugen.

Kameras können per Klick auf ein Objekt zentriert werden und folgen fortan dessen Position in einem festgelegten Abstand. Über das Mauselement kann der Nutzer an das Objekt heranzoomen. Hierbei wird nicht der Abstand zwischen Kamera und Objekt verringert, sondern lediglich der „Field of View“-Parameter. Das Rotieren der Kamera um ein Objekt ist bisher nicht möglich.

Befinden sich Moleküle an einer Kathode, werden ihre jeweiligen Ladungen, sofern definiert, richtig angezeigt. Wurde eine hinreichende Anzahl von Protonen in der Szene erzeugt, entstehen aus einem CO_2 -Molekül ein Methan- und zwei Wassermoleküle. Befinden sich darüber hinaus noch andere CO_2 -Moleküle an der Kathode, kommt es abhängig vom Abstand zwischen diesen auch zur Bildung von Ethen-Molekülen. Die auf Abbildung 2.1 dargestellte Testreaktion läuft damit korrekt ab.

5.1.2. Testszenarien

In den ersten vier Testszenarien werden nur halb so viele Protonen in der Szene instanziiert, wie für den vollständigen Reaktionsablauf aller CO_2 -Moleküle notwendig wären. Außer bei vielen instanziierten Molekülen ist zu beobachten, dass kein CO_2 -Molekül vollständig in Methan oder Ethen umgesetzt wird - da nicht genügend Protonen zur Verfügung stehen, bilden sich nur verschiedene Zwischenprodukte.

In den nächsten vier Testszenarien werden exakt so viele Protonen in der Szene instanziiert, wie für den vollständigen Reaktionsablauf aller CO_2 -Moleküle notwendig wären. Die Anzahl der entstandenen Methan-Moleküle steigt sofort an. Bei vielen in der Szene instanziierten Molekülen und Protonen erfolgt die Reaktion sehr schnell; hier muss der Nutzer auf die Zeitleiste zurückgreifen, um den Überblick zu behalten.

Nr.	CO ₂	H ⁺	Zeit (s) bis zur Entstehung von Methan oder Ethen	Anzahl Methan und Ethen nach einer Minute
1	5	20	-	-
2	10	40	-	-
3	20	80	-	-
4	40	160	2.7s, CH ₄	1 CH ₄
5	5	40	8.5s, CH ₄	3 CH ₄
6	10	80	11.9s, CH ₄	6 CH ₄
7	20	160	1.6s, CH ₄	8 CH ₄
8	40	320	1.3s, CH ₄	13 CH ₄ , 3 C ₂ H ₄
9	5	80	3.7s, CH ₄	4 CH ₄
10	10	160	2.5s, CH ₄	10 CH ₄
11	20	320	1.5s, CH ₄	18 CH ₄
12	40	640	0.4s, CH ₄	33 CH ₄ , 1 C ₂ H ₄

Tabelle 5.2.: Analyse von zwölf Testszenarien mit Spawn-Punkt-Koordinaten bei (15,15,15) und Würfel-Verteilung (Seitenlänge jeweils 25 Einheiten)

Bei 40 instanziierten Molekülen und 320 Protonen kommt zum ersten Mal in der Testreihe Ethen als Reaktionsprodukt vor. Dies liegt darin begründet, dass die Moleküle sich an der Kathode nah beieinander befinden. Die Wahrscheinlichkeit, dass zwei Moleküle miteinander reagieren können, bevor ein Proton auf diese trifft, ist somit höher.

In den nächsten vier Testszenarien werden doppelt so viele Protonen in der Szene instanziiert, wie für den vollständigen Reaktionsablauf aller CO₂-Moleküle notwendig wären. Die Reaktion läuft innerhalb einer geringeren Zeitspanne ab und ein Großteil der CO₂-Moleküle durchläuft die gesamte Reaktionskette. Bei 40 Instanzen entsteht hier allerdings nur ein einziges Mal Ethen. Die Moleküle befinden sich zwar nah beieinander, kollidieren aber im Millisekundentakt mit Protonen und reagieren so zu Methan weiter.

Die Ergebnisse weiterer getesteter Szenarien sind dem Anhang unter A.2 beigelegt. Die Position des Spawn-Punkts wurde hierbei angepasst, um zu analysieren, inwiefern dessen Platzierung sich auf die Geschwindigkeit und den Ablauf der Reaktion auswirkt.

5.2. Identifizierte Probleme

Ausgehend von der Beobachtung des Reaktionsverlaufs lassen sich Probleme verschiedener Art identifizieren, auf die im Folgenden näher eingegangen wird.

5.2.1. Visualisierung

Obwohl der Reaktionsmechanismus korrekt abläuft, ist die Reaktionsvisualisierung nicht optimal. Bei normaler Abspielgeschwindigkeit und einer hohen Anzahl instanzierter Protonen erfolgt die Reaktion so schnell, dass der Nutzer die einzelnen Zwischenschritte nicht nachvollziehen kann. Werden dagegen nur wenige Protonen instanziiert, kann es vorkommen, dass der Nutzer eine Weile warten muss, bis eines davon auf ein geladenes Molekül trifft und mit diesem weiterreagiert.

Befinden sich zahlreiche geladene Moleküle an der Kathode, können sich deren Ladungsanzeigen so überlappen, dass der angezeigte Wert nicht mehr lesbar ist. Allgemein steigt bei vielen CO_2 -Molekülen in der Szene zwar die Wahrscheinlichkeit, dass sämtliche Reaktionsschritte in kurzer zeitlicher Abfolge beobachtet werden können, es leidet aber die Übersichtlichkeit.

5.2.2. Kollisionserkennung

Trotz der kontinuierlichen, dynamischen Kollisionserkennung kommt es selten vor, dass Kollisionen bei hoher Geschwindigkeit nicht registriert werden. Besonders störend wirkt sich dieser Umstand dann aus, wenn die nicht registrierte Kollision zwischen einer Würfel-Plane und einem Molekül bzw. Proton hätte stattfinden müssen, die jeweilige Komponente aber aus dem Würfel ausbricht. Damit kann dem Nutzer potenziell auch eine Kamera verloren gehen, wenn diese auf die entflohene Komponente zentriert ist und das Sichtfeld sich nicht mehr so weit vergrößern lässt, als dass andere, raycastbare Komponenten als Kamera-Referenzobjekt angeklickt werden können.

5.2.3. Physikalische Verhaltensweisen

Ein weiteres Problem, die Festlegung der Eigenschaften von Protonen-Objekten, hängt mit der Physikberechnung in *Unity* zusammen. Protonen sind wie bereits beschrieben als Kugeln dargestellt, die etwas kleiner als ein Wasserstoffatom-Modell ausfallen. Physikalisch korrekt wäre es, wenn auch die Masse eines Proton-Objekts in etwa der des genannten Modells entspräche. In Verbindung mit der *Collider*-Komponente eines Proton-Objekts kommt es hierbei jedoch zu unerwünschtem physikalischem Verhalten: Wird ein Proton von einer Kathode beschleunigt und trifft auf ein Molekül, welches sich an dieser befindet und bereits über eine Ladung verfügt, bewegt sich das Molekül nach dem Zusammenstoß abhängig von seiner Masse, dem Impuls des Protons, dem Punkt der Kollision sowie der *PhysicMaterial*-Eigenschaften beider Komponenten (Elastizität, Reibung). Zwar ist dies rein physikalisch korrekt, kann aber dazu führen, dass durch die Krafteinwirkung chemische Verbindungen, die eigentlich instabil sind und so nur an der Kathode existieren, von dieser weggeschleudert werden.

5.2.4. Performanz

Der Test-Würfel bietet (abhängig von deren Größe) mehreren hundert Instanzen Platz, es können in jedem Fall so viele Komponenten darin erzeugt werden, um die Beispielreaktion darin ablaufen zu lassen. Jedoch ist zu erwarten, dass die Performanz durch eine Vergrößerung des Test-Würfels und die Generierung einer hohen Anzahl an Komponenten beeinträchtigt wird. Zum einen haben die in AtomCAD enthaltenen 3D-Modelle dafür, dass es sich nur um 3D-Grundformen handelt, einen hohen Polycount: Eine Atomkugel besteht in Maya aus 256 Flächen. Bei den meisten davon handelt es sich um Vierecke, die von *Unity* zusätzlich in zwei Dreiecke tesseliert werden. Das Methan-Modell enthält bereits in Maya 1280 Flächen. Der hohe Polycount wurde gewählt, damit die Modelle auch bei näherer Betrachtung möglichst rund erscheinen.

Weiterhin wirken sich verschiedene *Update*- und *FixedUpdate*-Funktionen in besonderem Maße belastend auf die Performanz aus. Ein Beispiel hierfür ist das Hinzufügen von Anziehungskräften, bei dem eine umfangreiche Abfrage der in der Szene befindlichen geladenen Komponenten erfolgen muss.

Obwohl sich AtomCAD zu einem großen Teil auf physikalische Simulation stützt und es in diesem Sinne irreführend scheinen kann, es als Animationstool zu bezeichnen, ist es aus den genannten Gründen nicht dafür ausgelegt, in größerem Maßstab für Simulationen genutzt zu werden. Viel eher erübrigt sich durch die physikalische Simulation die Notwendigkeit, das Verhalten von Komponenten animieren zu müssen. So entsteht eine Methode der effektiven Visualisierung im kleinen Maßstab, welcher für die Veranschaulichung von Reaktionsabläufen am Besten geeignet scheint.

6. Zusammenfassung

Ziel dieser Arbeit war die prototypische Entwicklung und Implementierung eines Tools, welches der Visualisierung katalytischer Vorgänge dient. Dazu wurde in Kapitel 2 untersucht, wie vergleichbare Anwendungen bezüglich ihrer Funktionen und ihres Designs aufgebaut sind. Kapitel 3 diente der Spezifizierung der an das Programm gestellten Anforderungen. Daraus entstand das Konzept für AtomCAD, ein simulationsgestütztes Animationstool zur Darstellung chemischer Reaktionen. Kapitel 4 befasste sich mit der Implementierung dessen prototypischer Benutzeroberfläche in der *Unity Engine*. Die Evaluation der Kernfunktionalität und die dabei identifizierten Probleme wurden schließlich in Kapitel 5 beschrieben.

Der Reaktionsablauf der elektrochemischen CO₂-Reduktion wurde mit AtomCAD vollständig visualisiert. Über das implementierte Graphical User Interface kann der Nutzer die benötigten Reaktionskomponenten definieren und instanziiieren. Die bewegliche Kamera ermöglicht das Beobachten der Reaktion von verschiedenen Positionen aus, während die Zeitleiste die Anpassung der Reaktionsgeschwindigkeit erlaubt.

6.1. Optimierungsmöglichkeiten

Auf Basis der im Abschnitt 5.2 identifizierten Probleme lassen sich zunächst einige Verbesserungsansätze für die aktuell implementierten Funktionen ableiten.

6.1.1. Kollisionserkennung

Für die in 5.2.2 beschriebenen Probleme kommen verschiedene Lösungsansätze infrage. Eine Möglichkeit besteht in der Definition einer Höchstgeschwindigkeit für Atome

und Moleküle. Diese müsste in den korrespondierenden *FixedUpdate*-Funktionen abgefragt und die Komponenten bei Bedarf gebremst werden. Auch denkbar wäre die Verkleinerung des Intervalls, in dem es zur Überprüfung von Kollisionen kommt oder die Vergrößerung der Objekt-*Collider*, welche allerdings zu unrealistischen Wechselwirkungen zwischen den Objekten führen würde [SW17, S.221f].

6.1.2. Physikalische Verhaltensweisen

Den in 5.2.3 aufgeführten Verhaltensweisen sollte nach Möglichkeit Abhilfe geschaffen werden, ohne auf *Rigidbody* und *Collider* von Proton-Objekten Einfluss zu nehmen. Ein Proton benötigt zwangsläufig einen *Collider*, welcher nicht als *Trigger* definiert ist, damit es zu physikalischen Interaktionen mit der Reaktionsumgebung kommen kann. Übrig bleibt eine Verringerung des Masse-Parameters, die allerdings physikalisch inkorrekt wäre und den unerwünschten Wechselwirkungen bei entsprechender Geschwindigkeit des Proton-Objekts keinen Abbruch täte.

Es scheint dagegen vielversprechender, einem Proton zwei *Collider* zuzuordnen. Einer davon wird auf die doppelte Größe des Modells skaliert und als *Trigger* definiert. Bewegt sich ein Proton nun in die Nähe eines geladenen Moleküls, können über den *Trigger* Skripts ausgelöst werden, bevor der eigentliche *Collider* des Protons mit dem des Moleküls in Kontakt tritt. Zwischen Proton-Objekten und anderen Proton-Objekten, nicht geladenen Molekülen sowie den Würfel-Flächen sind so noch immer Kollisionen möglich, bei denen physikalische Kräfte wirken.

6.2. Ausblick

Es lassen sich zahlreiche Ansätze für Erweiterungen identifizieren. Ein Teil davon betrifft die Inklusion weiterer Reaktionsparameter und Reaktionskomponenten, aber auch für eine Vergrößerung des Funktionsumfangs sind viele Möglichkeiten geboten.

6.2.1. Parameter

Um den Reaktionsverlauf in der Reaktionsumgebung abhängiger von einer höheren Anzahl von Eingaben des Nutzers zu gestalten, könnten eine Reihe von zusätzlichen Reaktionsparametern eingebaut werden. Im Anwendungsbeispiel ist zunächst davon ausgegangen worden, dass die angelegte Spannung simultan mit dem Verlauf der Zeit erhöht wird, das benötigte Einsatzpotenzial der jeweiligen elektrochemischen Teilreaktion in jedem Fall vorhanden ist und die Reaktion bis zur Bildung von Methan bzw. Ethen weiterlaufen kann.

In der Realität gibt es für jede Verbindung eine bestimmte Mindestspannung, welche Voraussetzung für das Auftreten der Reaktion ist. Diese entspricht jedoch nur den thermodynamischen Gleichgewichtsbedingungen; aufgrund von kinetischen Hemmungen an den Elektroden muss häufig eine deutlich höhere Spannung (Überspannung) an diese angelegt werden. Wissenschaftliche Erkenntnisse liegen dazu noch nicht vor, deshalb wurde im Rahmen dieser Arbeit mit der Vereinfachung gearbeitet, diese Überspannungen zu vernachlässigen und eine gleichmäßige sequentielle Abfolge der Teilreaktionen zu ermöglichen. In Erweiterungen des Programms sollen Erkenntnisse aus wissenschaftlichen Arbeiten über auftretende Überspannungen für die jeweiligen Teilreaktionen einfließen bzw. der Nutzer selbst solche Eingaben machen können.

6.2.2. Komponenten

Ein zentraler Aspekt der geplanten Erweiterungen ist die Generierung von Molekülen auf Basis von XML-Dateien, wodurch sich das Modellieren neuer Verbindungen in einem 3D-Programm erübrigen würde. Die Herangehensweise sollte hier allerdings anders ausfallen als bei der Generierung von Atom-Objekten, deren Eigenschaften in einer Textdatei festgelegt sind, die von *Unity* lediglich weiterverarbeitet wird.

Ist ein geeignetes Format für die Beschreibung von Molekülen erst einmal gefunden, dürfte es auch als Laie möglich sein, mit dieser Vorlage weitere Moleküle in XML zu definieren. Dies jedoch bricht die Immersion des Nutzers. Viel eher bietet es sich an, das Format nicht nur als Schreib-, sondern auch als Speichermuster zu verwenden. Der Aufbau der XML-Datei bleibt hierbei gleich; Nutzer könnten, sofern erwünscht, Moleküle schriftlich ergänzen. Die intuitivere Variante wäre jedoch,

den in 4.2.4 beschriebenen Baukasten zu utilisieren, um Moleküle zu entwerfen und deren Eigenschaften in eben dem Format abzuspeichern, auf dessen Basis wieder die Instanziierung erfolgt. So könnte der Nutzer sich seine eigene Molekül-Datenbank aufbauen, ohne die Gültigkeit der XML-Datei zu gefährden.

Mit der Implementierung des Elektroden-Dropdowns wurde bereits die Grundlage für eine einfache Ergänzung der Elektrodenmaterialien geschaffen. Bisher beschränkt sich der Unterschied zwischen den Materialien jedoch lediglich auf deren Anzeigefarbe. In der Realität bestimmt das Elektrodenmaterial den Ablauf und die Produkte der elektrochemischen CO₂-Reduktion. An Silberkathoden entstehen in etwa andere Verbindungen als an Kupferkathoden. Damit die Reaktion davon abhängig korrekt in der Reaktionsumgebung abläuft, könnte beispielsweise die Reaktions-Klasse um einen zusätzlichen Elektrodenmaterial-Parameter ergänzt werden. Außerdem müsste ein Entwickler entweder den Molekül-Baukasten implementieren oder neue Modelle in einem 3D-Programm erstellen. Die Auswahl der Moleküle ist bisher auf die Verbindungen limitiert, die beim von Nørskov und Koper vorhergesagten Reaktionsmechanismus auftreten (siehe Abbildung 2.1).

Ein weiterer Aspekt ist die Darstellung von Elektroden, die bisher nur als gleichmäßige Metallgitterstruktur erfolgt. Dies bringt dann Schwierigkeiten mit sich, wenn die Materialstruktur der Elektrode sich entscheidend auf deren Funktionsweise auswirkt. Bei den bereits implementierten Elektroden konnte davon ausgegangen werden, dass die Übertragung von Elektronen unabhängig von der genauen Position des Kontakts stattfindet. Dies trifft jedoch nicht auf alle für die elektrochemische CO₂-Reduktion infrage kommenden Elektrodenmaterialien zu. Nach effektiven und effizienten Elektrodenmaterialien wie beispielsweise Porphyrinen wird beständig geforscht [RBJ⁺17]. Porphyrine zeichnen sich dadurch aus, dass ein Metallion in einer Chelat-umgebung stabil eingebunden ist (siehe Abbildung 6.1). Es wird vermutet, dass die Anbindung des CO₂-Moleküls über das Metallion und die umliegenden Stickstoffe des Tetrapyrrolrings erfolgt. Der Abstand zwischen diesen aktiven Zentren bestimmt die Reaktionsprodukte. Mit einer entsprechenden Gitterstruktur ließe sich dieses Verhalten zwar simulieren, indem den einzelnen Atom-Modellen selektiv Skripts zugewiesen werden; die automatische Instanziierung einer derartigen Oberfläche wäre jedoch um einiges schwieriger.

Bei einer Erweiterung ebenfalls zu beachten ist die genaue Unterscheidung zwischen Katalysatoren und Elektroden. Obwohl beide Begriffe in dieser Arbeit synonym

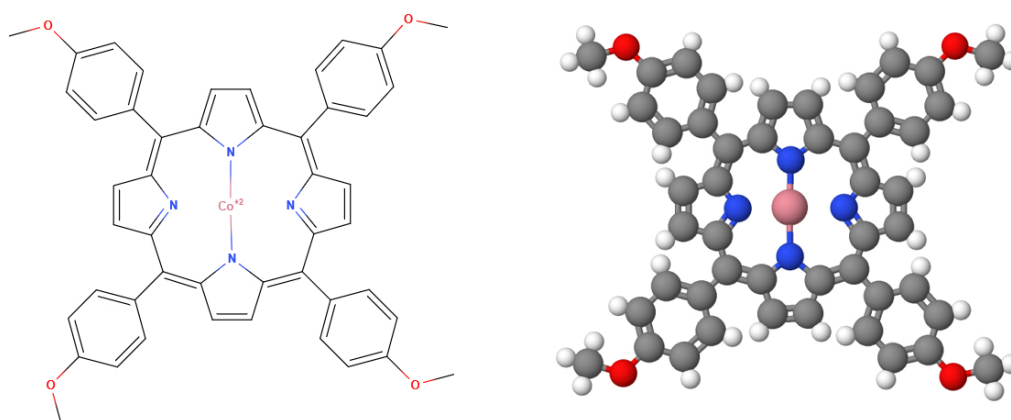


Abbildung 6.1.: Darstellung von Cobalt-Tetramethoxyphenylporphyrin mittels Valenzstrichformel, in *MolView* daraus generiertes 3D-Modell. Über das Cobaltion in der Mitte sowie die umliegenden Stickstoffe (blau gekennzeichnet) könnte die Anbindung von CO_2 erfolgen.

zueinander verwendet werden können, ist weder jede Elektrode ein Katalysator noch jeder Katalysator eine Elektrode. Im Anwendungsfall wirken die Elektroden katalytisch; sie ermöglichen den Verlauf der Reaktion, ohne selbst dabei verbraucht zu werden. Solange zwischen Katalysatoren und Elektroden jedoch nicht differenziert wird, dient AtomCAD nur der Visualisierung elektrochemischer Vorgänge anstatt katalytischer Reaktionen im Allgemeinen.

6.2.3. Funktionen

Die Implementierung zusätzlicher Parameter und Komponenten bildet die Grundlage für eine Weiterentwicklung von AtomCAD. Ausgehend davon entstehen zahlreiche Möglichkeiten, den Funktionsumfang des Programms zu vergrößern.

Besondere Beachtung sollte an dieser Stelle dem Molekül-Baukasten zukommen, dessen Implementierung der Ausgangspunkt für die dynamische Generierung von Molekülen wäre. Eine daraus ableitbare Funktion ist das Im- und Exportieren von Verbindungen. Hier müsste ein gängiges Format zur Beschreibung chemischer Daten wie etwa die Chemical Markup Language (CML) gewählt werden. So könnte der Nutzer Verbindungen aus fachspezifischen Datenbanken wie der Protein Data Bank

[Wor17] herunterladen und in AtomCAD einbinden. Wird dann noch die Möglichkeit geschaffen, Reaktionen für Moleküle direkt im User Interface festzulegen, könnte sich aus AtomCAD ein vielseitiges Visualisierungstool für chemische Reaktionen entwickeln, welches keinesfalls auf katalytische Vorgänge beschränkt wäre.

Literaturverzeichnis

- [Adv17] Advanced Chemistry Development, Toronto: *ACD/ChemSketch Reference Manual*, 2017.
- [Ber15] Bergwerf, Herman: *MolView Help*, 2015, URL: <http://molview.org/>, besucht am 10.12.2017.
- [Bun13] Bundesministerium für Bildung und Forschung (Hg.): *Technologien für Nachhaltigkeit und Klimaschutz - Chemische Prozesse und stoffliche Nutzung von CO₂*, Bonn, 2013, URL: https://www.bmbf.de/pub/Technologien_fuer_Nachhaltigkeit_und_Klimaschutz.pdf, besucht am 10.12.2017.
- [CVK13] Federico Calle-Vallejo und Marc T. M. Koper: *Theoretical Considerations on the Electroreduction of CO to C₂Species on Cu(100) Electrodes*, *Angewandte Chemie*, Bd. 125(28):S. 7423–7426, 2013, URL: <https://doi.org/10.1002/ange.201301470>, besucht am 10.12.2017.
- [Das17] Dassault Systèmes, San Diego: *BIOVIA Draw Help*, 2017.
- [DIN98] DIN, Deutsches Institut für Normung (Hg.): *Bildschirmarbeitsplätze. 1, Arbeitsplatz und Lichttechnik*, Bd. 194 von *DIN-Taschenbuch*, Beuth, Berlin, 4. Aufl., 1998.
- [Dri15] Matthias Drieß: *Die künstliche Photosynthese: Von der Vision zur Realisierung*, Videoaufnahme zur Vorlesung, 2015, URL: <http://www.uni-heidelberg.de/archiv/18492>, besucht am 10.12.2017.
- [Erl03] Helmut Erlenkötter: *XML - Extensible Markup Language von Anfang an*, Rowohlt-Taschenbuch, Reinbek bei Hamburg, 3. Aufl., 2003.
- [Fri17] Christian Frings: *Gestaltgesetze, Gestaltfaktoren*, in *Dorsch - Lexikon der Psychologie* (herausgegeben von Markus A. Wirtz), Hogrefe,

- Bern, 18 Aufl., 2017, URL: <https://m.portal.hogrefe.com/dorsch/gestaltgesetze-gestaltfaktoren/>, besucht am 10.12.2017.
- [Geh14] Daniel Gehrer: *CellUnity an Interactive Tool for Illustrative Visualization of Molecular Reactions*, Bachelorarbeit, Technische Universität Wien, Institut für Computergraphik und Algorithmen, 2014, URL: https://www.cg.tuwien.ac.at/research/publications/2014/Gehrer_Daniel_CUI/, besucht am 10.12.2017.
- [GI08] E. Bruce Goldstein und Hans Irtel (Hg.): *Wahrnehmungspsychologie. Der Grundkurs*, Springer Spektrum, Berlin, 7. Aufl., 2008.
- [Har15] Volker Hartmann: *Die Photosynthese als erneuerbare Energie - Zukünftige Produktion von Biowasserstoff aus Sonnenlicht*, Springer Spektrum, Wiesbaden, 1. Aufl., 2015, URL: <https://doi.org/10.1007/978-3-658-09187-3>, besucht am 10.12.2017.
- [JM17] Jens Jacobsen und Lorena Meyer: *Praxisbuch Usability und UX*, Rheinwerk, Bonn, 1. Aufl., 2017.
- [Jmo17] Jmol: an open-source Java viewer for chemical structures in 3D: *Colors*, 2017, URL: http://jmol.sourceforge.net/jscolors/#color_H, besucht am 10.12.2017.
- [Knu17] Kerstin Knura: *Entwicklung und Implementierung eines virtuellen Labors zur Durchführung biochemischer Experimente am Beispiel einer Verdünnungsreihe mittels Pipettieren*, Bachelorarbeit, Hochschule Mittweida, Fakultät Angewandte Computer- und Biowissenschaften, 2017.
- [Kol65] Walter L. Koltun: *Precision space-filling atomic models*, *Biopolymers*, Bd. 3(6):S. 665–679, 1965, URL: <http://dx.doi.org/10.1002/bip.360030606>, besucht am 10.12.2017.
- [Kri17] Friedhelm Kring: *Die Grundsätze der Dialoggestaltung nach ISO 9241-110*, 2017, URL: <https://www.weka-manager-ce.de/betriebsanleitung/grundsaeetze-dialoggestaltung-iso-9241-110/>, besucht am 10.12.2017.

- [KSHY15] Dohyung Kim, Kelsey K. Sakimoto, Dachao Hong und Peidong Yang: *Künstliche Photosynthese für die Produktion von nachhaltigen Kraftstoffen und chemischen Produkten*, *Angewandte Chemie*, Bd. 127(11):S. 3309–3316, 2015, URL: <http://dx.doi.org/10.1002/ange.201409116>, besucht am 10.12.2017.
- [Lab17] Labster: *Simulations*, 2017, URL: <https://www.labster.com/simulations/>, besucht am 10.12.2017.
- [LHZQ14] Yuyu Liu, Feng Hong, Jiuju Zhang und Jinli Qiao: *A review of catalysts for the electroreduction of carbon dioxide to produce low-carbon fuels*, *Chemical Society Reviews*, Bd. 43(2):S. 631–675, 2014, URL: <http://dx.doi.org/10.1039/c3cs60323g>, besucht am 10.12.2017.
- [Max17] Max-Planck-Institut für Eisenforschung GmbH: *Elektrochemische CO₂-Reduktion – mit Hochdurchsatz auf der Suche nach neuen Katalysatoren*, 2017, URL: <https://www.mpie.de/2973650/Electrochemical-CO2-reduction>, besucht am 10.12.2017.
- [MEL17] MEL Science: *MEL Chemistry VR*, 2017, URL: <https://melscience.com/vr/>, besucht am 10.12.2017.
- [Nie93] Jakob Nielsen: *Usability Engineering*, Academic Press Professional, Boston, 1. Aufl., 1993.
- [NM94] Jakob Nielsen und Robert L. Mack: *Usability Inspection Methods*, John Wiley & Sons, New York, 1. Aufl., 1994.
- [NPAP⁺10] Jens K. Nørskov, Andrew A. Peterson, Frank Abild-Pedersen, Felix Studt und Jan Rossmeisl: *How copper catalyzes the electroreduction of carbon dioxide into hydrocarbon fuels*, *Energy & Environmental Science*, Bd. 3(9):S. 1311–1315, 2010, URL: <https://doi.org/10.1039/C0EE00071J>, besucht am 10.12.2017.
- [Pre01] Marc Prensky: *Digital Natives, Digital Immigrants Part 1, On the Horizon*, Bd. 9(5):S. 1–6, 2001, URL: <http://www.emeraldinsight.com/doi/abs/10.1108/10748120110424816>, besucht am 10.12.2017.
- [RBJ⁺17] Jan Rossmeisl, Alexander Bagger, Wen Ju, Ana S. Varela und Peter Strasser: *Single site porphyrine-like structures advantages over*

- metals for selective electrochemical CO₂ reduction*, *Catalysis Today*, Bd. 288:S. 74–78, 2017, URL: <https://doi.org/10.1016/j.cattod.2017.02.028>, besucht am 10.12.2017.
- [RCS17] RCSB PDB: *Molecular Graphics Software Links*, 2017, URL: http://www.rcsb.org/pdb/static.do?p=software/software_links/molecular_graphics.html, besucht am 10.12.2017.
- [SVKR16] Peter Strasser, Ana S. Varela, Matthias Kroschel und Tobias Reier: *Controlling the selectivity of CO₂ electroreduction on copper: The effect of the electrolyte concentration and the importance of the local pH*, *Catalysis Today*, Bd. 260:S. 8–13, 2016, URL: <https://doi.org/10.1016/j.cattod.2015.06.009>, besucht am 10.12.2017.
- [SW17] Carsten Seifert und Jan Winslaug: *Spiele entwickeln mit Unity 5 - 2D- und 3D-Games mit Unity und C# für Desktop, Web & Mobile. Für Unity 5.6*, Carl Hanser, München, 3. Aufl., 2017.
- [Uni17a] Unity Technologies: *Scripting API 2017.2*, 2017, URL: <https://docs.unity3d.com/ScriptReference/index.html>, besucht am 10.12.2017.
- [Uni17b] Unity Technologies: *Unity Manual 2017.2*, 2017, URL: <https://docs.unity3d.com/Manual/index.html>, besucht am 10.12.2017.
- [Wor17] Worldwide Protein Data Bank: *wwPDB Frequently Asked Questions*, 2017, URL: <http://www.wwpdb.org/about/faq>, besucht am 10.12.2017.

Anhang

Anhangsverzeichnis

Anhang - Abbildungs- und Tabellenverzeichnis	XVII
A Analysedokumente	A1
A.1 Reaktionsvisualisierung	A1
A.2 Reaktionsevaluation	A5
B Konzeptionsdokumente	A7
B.1 Visualisierung	A7
B.2 Usability	A11
B.3 Programmierung	A12

Anhang - Abbildungs- und Tabellenverzeichnis

A.1.1	Darstellung bereits implementierter Elektrodenmaterialien in der Reaktionsumgebung	A1
A.1.2	Instanziierung verschiedener Atom-Modelle in der Reaktionsumgebung	A2
A.1.3	Verteiler-Resultate beim Instanziieren von H ₂ O-Molekülen	A3
A.1.4	Reaktionsmechanismus der elektrochemischen CO ₂ -Reduktion in AtomCAD	A4
A.2.1	Analyse von zwölf Testszenarien (2)	A5
A.2.2	Analyse von zwölf Testszenarien (3)	A6
B.1.1	Gestaltprinzipien	A8
B.1.2	Darstellung von Ameisensäure in <i>MolView</i>	A9
B.1.3	Bereits implementierte <i>Materials</i> auf Basis des <i>Jmol</i> -Farbmodells	A10
B.2.1	Die sieben Grundsätze der Dialoggestaltung	A11
B.3.1	Im Projektordner enthaltene C#-Klassen	A14

A. Analysedokumente

A.1. Reaktionsvisualisierung

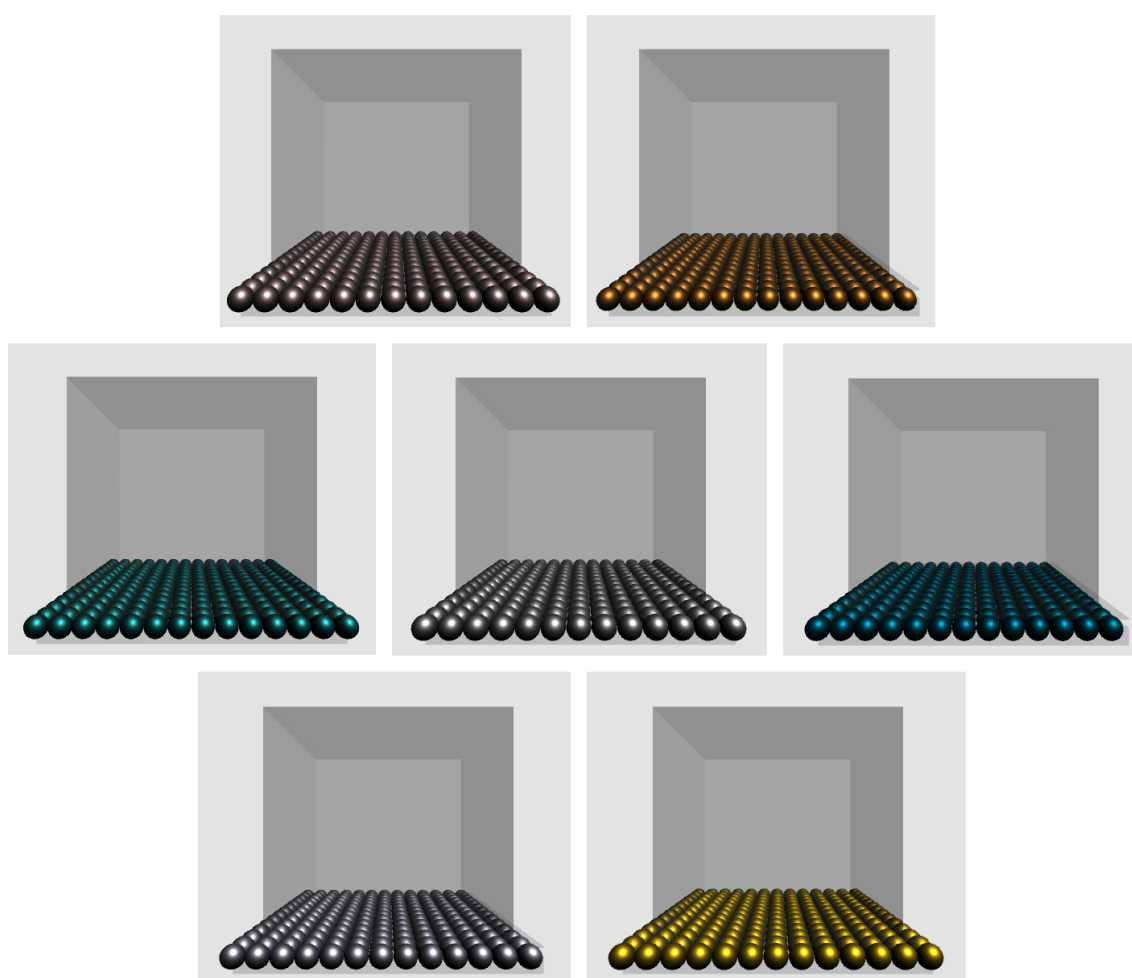


Abbildung A.1.1.: Darstellung bereits implementierter Elektrodenmaterialien in der Reaktionsumgebung: Aluminium, Kupfer, Ruthenium, Silber, Palladium, Platin und Gold.

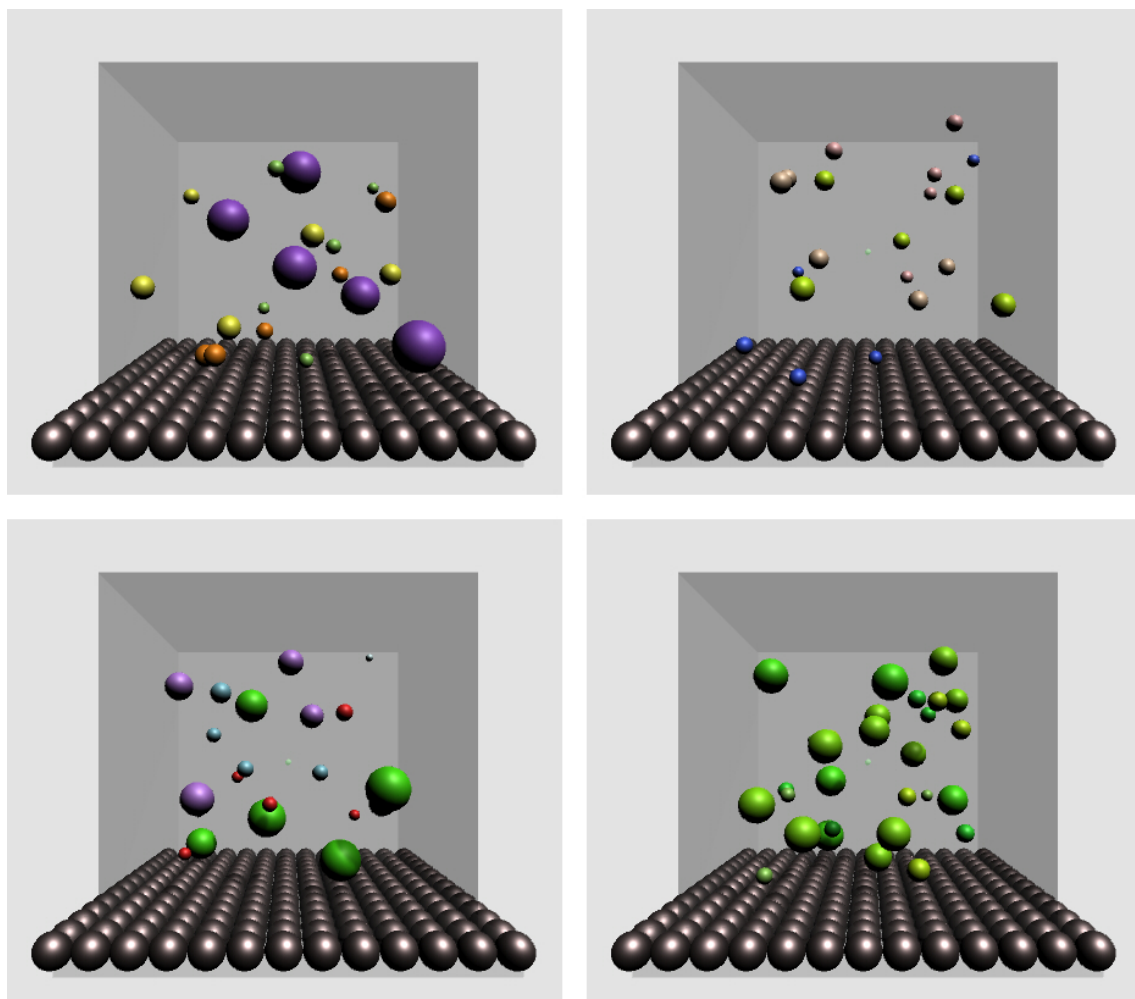


Abbildung A.1.2.: Instanziierung verschiedener Atom-Modelle in der Reaktionsumgebung. Die *Materials* sowie die Größe der Kugeln indizieren, dass deren Attribute korrekt aus der Periodensystem-XML-Datei gelesen werden.

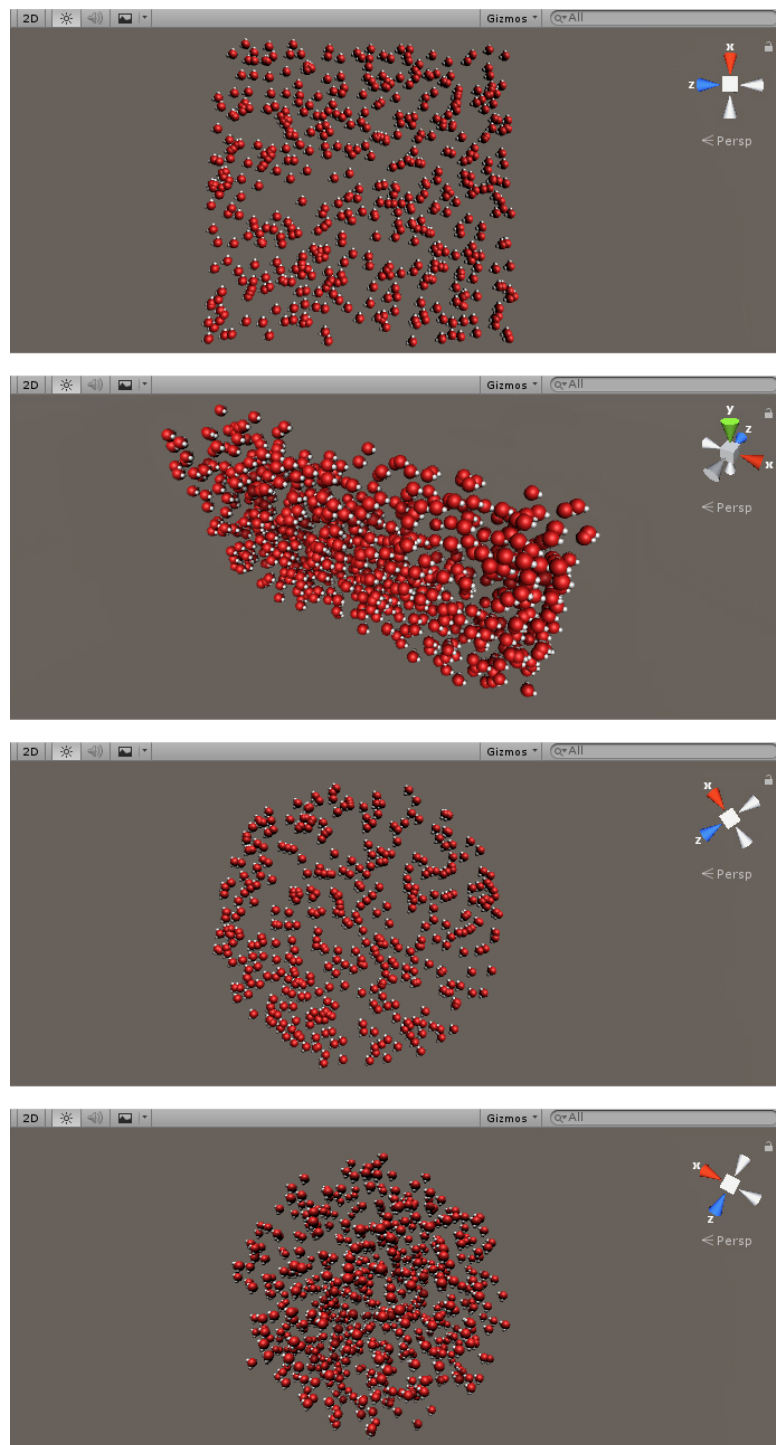


Abbildung A.1.3.: Verteiler-Resultate beim Instanzieren von H_2O -Molekülen: Rechteck, Quader, Kreis und Kugel

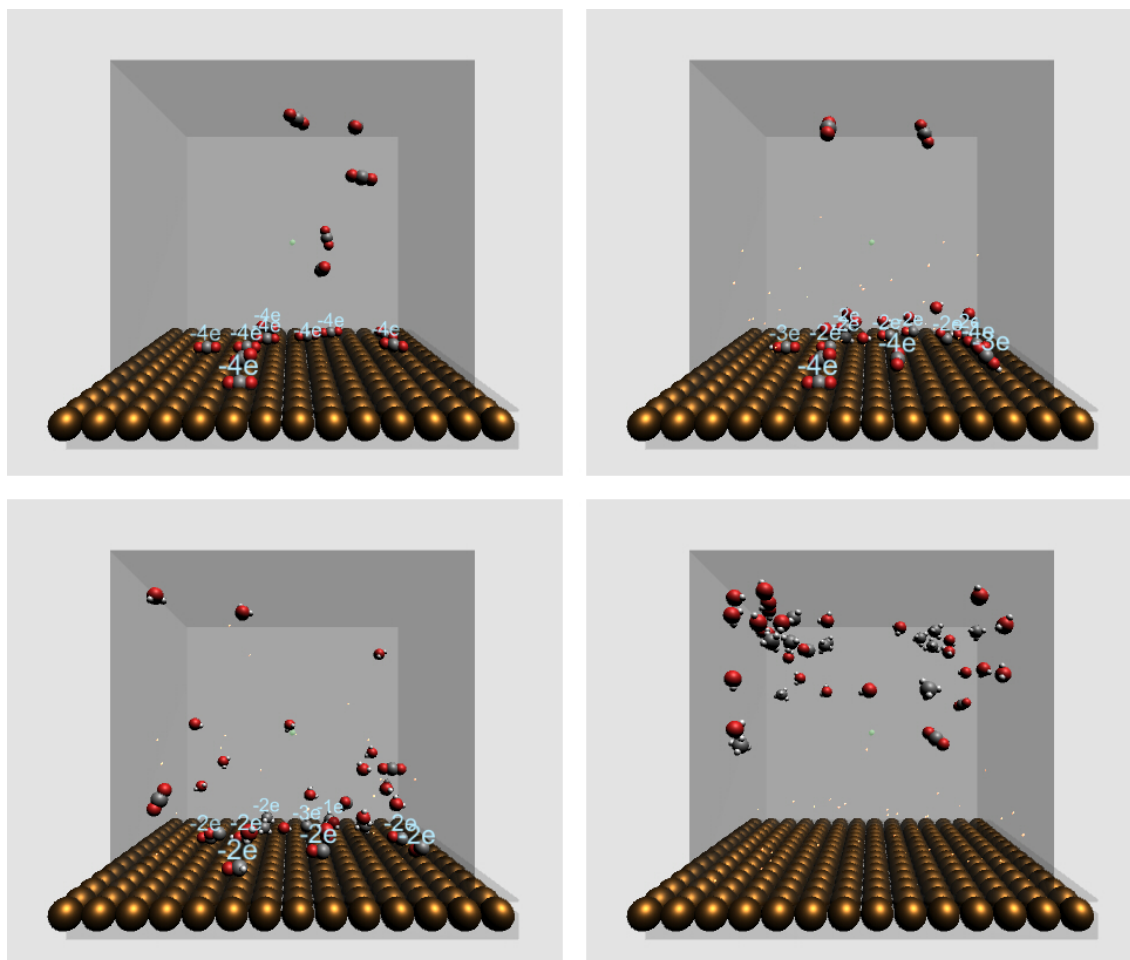


Abbildung A.1.4.: Reaktionsmechanismus der elektrochemischen CO₂-Reduktion an einer Kupferkathode in AtomCAD. Deutlich zu sehen ist die Überlappung von Ladungsanzeigen. Je mehr CO₂-Moleküle im Würfel instanziiert werden, desto unübersichtlicher wird die Visualisierung der Reaktion. Gleichzeitig erhöht eine hohe Anzahl an CO₂-Molekülen die Chance, dass Ethen als Reaktionsprodukt auftritt.

A.2. Reaktionsevaluation

Nr.	CO ₂	H ⁺	Zeit (s) bis zur Entstehung von Methan oder Ethen	Anzahl Methan und Ethen nach einer Minute
1	5	20	-	-
2	10	40	-	-
3	20	80	1.2s, CH ₄	2 CH ₄
4	40	160	0.8s, C ₂ H ₄	4 CH ₄ , 1 C ₂ H ₄
5	5	40	2.1s, CH ₄	2 CH ₄
6	10	80	4.1s, CH ₄	3 CH ₄
7	20	160	0.1s, CH ₄	9 CH ₄
8	40	320	0.1s, CH ₄	14 CH ₄ , 2 C ₂ H ₄
9	5	80	2.0s, CH ₄	5 CH ₄
10	10	160	0.1s, CH ₄	6 CH ₄ , 1 C ₂ H ₄
11	20	320	0.1s, CH ₄	19 CH ₄
12	40	640	?	?

Tabelle A.2.1.: Analyse von zwölf Testszenarien mit Spawn-Punkt-Koordinaten bei (5,5,5) ohne Verteiler. Die Reaktion verläuft schneller als in den ersten zwölf Testszenarien, da der Spawn-Punkt nah an der Kathode liegt und sich dort direkt nach deren Generierung viele Moleküle anbinden. Bei zu vielen instanziierten Komponenten stürzt *Unity* ab.

Nr.	CO ₂	H ⁺	Zeit (s) bis zur Entstehung von Methan oder Ethen	Anzahl Methan und Ethen nach einer Minute
1	5	20	30.8s, CH ₄	1 CH ₄
2	10	40	3.6s, CH ₄	1 CH ₄
3	20	80	5.2s, CH ₄	1 CH ₄
4	40	160	-	-
5	5	40	16.7s, CH ₄	2 CH ₄
6	10	80	10.9s, CH ₄	4 CH ₄
7	20	160	2.0s, CH ₄	12 CH ₄
8	40	320	0.1s, CH ₄	13 CH ₄
9	5	80	6.9s, CH ₄	5 CH ₄
10	10	160	2.5s, CH ₄	10 CH ₄
11	20	320	0.1s, CH ₄	19 CH ₄
12	40	640	?	?

Tabelle A.2.2.: Analyse von zwölf Testszenarien mit Spawn-Punkt-Koordinaten bei (15,25,15) ohne Verteiler. Die Reaktion verläuft ähnlich wie bei den ersten zwölf Testszenarien, da eine zufällige Verteilung der Moleküle im Würfel erfolgen kann, bevor diese sich an die Kathode binden. Bei zu vielen instanziierten Komponenten stürzt *Unity* ab. Grund dafür ist wahrscheinlich die zeitgleiche Instanziierung zahlreicher Protonen an einer Stelle, die zu einer hohen Anzahl zeitgleicher Kollisionen und damit *OnTriggerEnter*-Aufrufen führt. Für diese Theorie spricht, dass der Absturz nur bei den Testszenarien geschieht, in denen kein Verteiler genutzt wurde.

B. Konzeptionsdokumente

B.1. Visualisierung


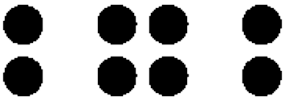


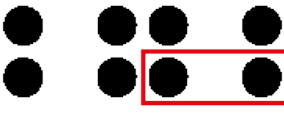


Prinzip	Beschreibung	Beispiel
Prägnanz	Jedes Objekt wird so wahrgenommen, dass die resultierende Struktur so einfach wie möglich ist	
Nähe	Objekte, die sich nah beieinander befinden, werden als Gruppe wahrgenommen	
Ähnlichkeit	Sich ähnelnde Objekte (Form, Farbe und Ausrichtung) werden als Gruppe wahrgenommen	
Symmetrie	Objekte werden eher als Gruppe wahrgenommen, wenn sie symmetrisch zueinander sind	
Region	Objekte in einem geschlossenen Bereich werden als Gruppe wahrgenommen	
Fortsetzung	Objekte werden als Gruppe wahrgenommen, wenn sie auf geraden oder kurvigen Linien angeordnet sein könnten	
Vertrautheit	Objekte werden eher als Gruppe wahrgenommen, wenn der Beobachter eine Bedeutung mit dieser assoziiert	

Tabelle B.1.1.: Eine Auswahl an Gestaltprinzipien, angelehnt an [JM17, S.46–49] sowie [GI08, S.108–113]

Modell	Atomradien	Bindungswinkel	Bindungsanzahl
 Valenzstrichformel			x
 Skelettmodell		x	x
 Stabmodell		x	
 Bälle & Stäbe	x	x	x
 CPK-Modell	x	x	

Tabelle B.1.2.: Darstellung von Ameisensäure mittels Valenzstrichformel, in *Mol-View* daraus generierte 3D-Modelle

Farbe	HEX-Farbwert	Element	Kürzel
	#FFFFFF	Wasserstoff	H
	#D9FFFF	Helium	He
	#CC80FF	Lithium	Li
	#C2FF00	Beryllium	Be
	#FFB5B5	Bor	B
	#909090	Kohlenstoff	C
	#3050F8	Stickstoff	N
	#FF0D0D	Sauerstoff	O
	#90E050	Fluor	F
	#B3E3F5	Neon	Ne
	#AB5CF2	Natrium	Na
	#8AFF00	Magnesium	Mg
	#BFA6A6	Aluminium	Al
	#F0C8A0	Silicium	Si
	#FF8000	Phosphor	P
	#FFFF30	Schwefel	S
	#1FF01F	Chlor	Cl
	#80D1E3	Argon	Ar
	#8F40D4	Kalium	K
	#3DFF00	Calcium	Ca
	#C88033	Kupfer	Cu
	#248F8F	Ruthenium	Ru
	#006985	Palladium	Pd
	#C0C0C0	Silber	Ag
	#D0D0E0	Platin	Pt
	#FFD123	Gold	Au

Tabelle B.1.3.: Bereits implementierte *Materials* auf Basis des *Jmol*-Farbmodells [Jmo17]. *Materials* von Kupfer bis Gold sind bisher nur für die Elektroden-Gitterstruktur verfügbar; Chemische Elemente wurden bis zur Ordnungszahl 20 (Calcium) in das XML-Periodensystem übernommen.

B.2. Usability

Eigenschaft	Beschreibung
Der Aufgabe angemessen	Die Anwendung leistet das, was der Benutzer von ihr erwartet. Sie unterstützt ihn und führt ihn schnell zum Ziel. Die eingesetzte Technik ist für den Nutzungsfall angemessen.
Selbstbeschreibend	Die Anwendung macht dem Nutzer deutlich, wie er sein Ziel erreicht und was er im jeweiligen Schritt tun soll. Eine klare Navigation und verständliche Anweisungen sind vorgegeben.
Steuerbar	Der Benutzer steuert die Anwendung, nicht umgekehrt. Prozesse können jederzeit abgebrochen werden, es gibt immer einen Weg zurück. Parameter wie die Lautstärke lassen sich regulieren.
Erwartungskonform	Die Anwendung verhält sich so, wie der Benutzer es erwartet. Dazu zählen Konsistenz innerhalb der Anwendung sowie die Berücksichtigung weit verbreiteter Konventionen.
Fehlertolerant	Das System kann mit falschen Benutzerinformationen umgehen und gibt bei Fehlern klare Rückmeldungen. Der Korrekturaufwand des Benutzers ist minimal.
Individualisierbar	Der Benutzer hat die Möglichkeit, die Anwendung an sein Vorwissen bzw. seine Vorlieben anzupassen.
Lernförderlich	Die Anwendung unterstützt den Benutzer dabei, den Umgang mit ihr schrittweise zu erlernen.

Tabelle B.2.1.: Die sieben Grundsätze der Dialoggestaltung nach ISO 9241-110, angelehnt an [JM17, S.61f] sowie [Kri17]

B.3. Programmierung

Klassenname(.cs)	Beschreibung
AtomMonoFunc	MonoBehaviour-Funktionalität für Atom-Objekte, kann die Objektparameter basierend auf Informationen überschreiben, die die Chelement-Klasse enthält
Chelement	Serialisierbare Klasse, dient der Beschreibung der von PeriodicTable aus einer XML-Datei gelesenen chemischen Elemente
Create	Helferklasse, erlaubt schnelles Instanzieren in der Testumgebung
Electrode	Serialisierbare Klasse, dient der Beschreibung der von ElectrodeContainer aus einer XML-Datei gelesenen Elektroden-Elemente. Grund der Markierung: Parameter reichen nicht aus, um die Eigenschaften komplexer Elektrodenoberflächen wie Porphyrinen zu beschreiben
ElectrodeContainer	Liest XML-Datei ein und speichert die einzelnen Elektroden-Elemente in einer Liste vom Typ Electrode, auf die ElectrodeManager zugreifen kann
ElectrodeManager	Singleton-Behaviour für Elektroden-Dropdown, ruft bei Änderungen Methode im Elektroden-Behaviour auf, die das Elektroden-Objekt mit neuen Attributen überschreibt
ElectrodeMatrixPart	MonoBehaviour-Funktionalität für einzelne Atom-Modelle eines Elektroden-Sammelobjekts
ElectrodeMonoFunc	MonoBehaviour-Funktionalität für Elektroden-Sammelobjekte, kann die Objektparameter basierend auf Informationen überschreiben, die die Electrode-Klasse enthält. Grund der Markierung: Siehe „Electrode“
LookAtCam	Wird Text Mesh eines Moleküls zugeordnet, bewirkt dessen Ausrichtung zur Kamera

MainMenuManager	Hauptmenü-Funktionalität. Grund der Markierung: Bisher keine Möglichkeit, im Optionen-Menü Anpassungen vorzunehmen; Beenden der Applikation wird erst bei einem Build funktionieren
Molecule	Serialisierbare Klasse, dient der Beschreibung der von MoleculeContainer aus einer XML-Datei gelesenen Molekül-Elemente
MoleculeContainer	Liest XML-Datei ein und speichert die einzelnen Moleküle in einer Liste vom Typ Molecule, auf die SpawnManager zugreifen kann
MoleculeMonoFunc	MonoBehaviour-Funktionalität für Molekül-Objekte, enthält alle nötigen Abfrageparameter und Kollisionsmethoden, die das Verhalten von Molekülen beeinflussen. Grund der Markierung: Verknüpfung mit der Molecule-Klasse ist noch nicht implementiert
PeriodicTable	Liest XML-Datei ein und speichert die einzelnen chemischen Elemente in einer Liste vom Typ Chelement, auf die SpawnManager zugreifen kann
PHManager	Singleton-Behaviour, dient dem Validieren des pH-Werts und ruft bei Änderungen entsprechende Methode im Molekül-Behaviour auf
Proton	Enthält <i>OnTriggerEnter</i> -Methode, die bei Zusammenstößen mit Molekülen die Protonenreaktion-Methode des Molekül-Behaviours auslöst
RaycastingManager	Singleton-Behaviour, kontrolliert die Bewegungen und Zoomfunktion von Kameras
Reaction	Serialisierbare Klasse, enthält Informationen zu den Voraussetzungen und Endprodukten einer Reaktion
SelectButton	Funktionalität für die Reaktions-Auswahl-Buttons im Auswahlmenü
SelectionMenuManager	Auswahlmenü-Funktionalität. Grund der Markierung: Speicher- und Ladefunktionen sind nicht implementiert, keine Generierung von Buttons zum Auswählen der Reaktionen

Semipermeability	Kann Flächen zugewiesen werden, kontrolliert Kollisionserkennung mit ausgewählten Objekten und lässt diese durch die Fläche passieren
ShowcaseMenuManager	Schaukasten-Funktionalität. Grund der Markierung: Speicher- und Ladefunktionen sind nicht implementiert, keine Generierung von Buttons zum Auswählen der Reaktionen
SpawnManager	Kontrolliert den Spawn-Vorgang anhand der in einem Spawn-Panel eingegebenen Parameter. Grund der Markierung: Keine ausreichende Validierung eingegebener Verteiler-Werte
SpawnPanel	Kontrolliert die Auswahl von Spawn-Panels, delegiert Nachrichten an Toggle-Groups
SpawnPanelManager	Singleton-Behaviour, kontrolliert das Generieren und Entfernen von Spawn-Panels im Spawn-Menü mithilfe von Position-Transform-Objekten
TagManager	Wird Kamera-Prefab zugewiesen, bewirkt das Tagging von Kameras als Hauptkamera, sobald sie aktiv geschaltet werden sowie das Entfernen dieser Kennzeichnung, sobald der Nutzer eine andere Kamera aktiviert. Unterstützt „LookAtCam“ dadurch, dass es immer nur ein als Hauptkamera markiertes Objekt geben kann, zu dem sich die TextMeshes ausrichten
TimeManager	Singleton-Behaviour, enthält Funktionalität für die Zeitleiste sowie einige Testfunktionen (Countdowns)
VoltageManager	Singleton-Behaviour, dient dem Validieren der Spannung und ruft bei Änderungen entsprechende Methode im Molekül-Behaviour auf

Tabelle B.3.1.: Im Projektordner enthaltene C#-Klassen. Rot: Nicht implementiert; Gelb: Unvollständig implementiert.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Mittweida, den 10. Dezember 2017

Anna Heinrich